Robust Surface Modelling of Visual Hull from Multiple Silhouettes

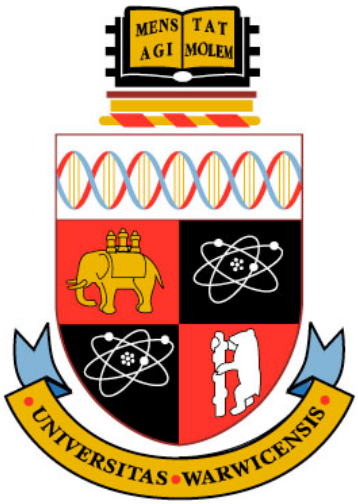
by

Dongjoe Shin, BSc, MSc

A thesis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

School of Engineering
University of Warwick



September 2008

Contents

List of Figures

List of Tables

Acknowledgements

I would like to take this opportunity to thank my research supervisor Dr. T. Tjahjadi for his academic and financial support towards this research. I would also like to express my gratitude to the annual progress panels in the School of Engineering: Dr. R. Staunton, Dr. N. Stocks, and Prof. J. Gardner. Their kind advice about the research was a constant help. Finally, I would like to thank my fellow students (and now Drs) for their support, generosity and encouragement.

Declaration

This thesis is submitted in partial fulfilment for the degree of Doctor of Philosophy under the regulations set out by the Graduate School at the University of Warwick. This thesis is solely composed of research undertaken by Dongjoe Shin under the supervision of Dr. Tardi Tjahjadi. The research materials have not been submitted in any previous application for a higher degree. All sources of information are specifically acknowledged in the content.

Abstract

Reconstructing depth information from images is one of the actively researched themes in computer vision and its application involves most vision research areas from object recognition to realistic visualisation. Amongst other useful vision-based reconstruction techniques, this thesis extensively investigates the visual hull (VH) concept for volume approximation and its robust surface modelling when various views of an object are available. Assuming that multiple images are captured from a circular motion, projection matrices are generally parameterised in terms of a rotation angle from a reference position in order to facilitate the multi-camera calibration. However, this assumption is often violated in practice, i.e., a pure rotation in a planar motion with accurate rotation angle is hardly realisable. To address this problem, at first, this thesis proposes a calibration method associated with the approximate circular motion.

With these modified projection matrices, a resulting VH is represented by a hierarchical tree structure of voxels from which surfaces are extracted by the Marching cubes (MC) algorithm. However, the surfaces may have unexpected artefacts caused by a coarser volume reconstruction, the topological ambiguity of the MC algorithm, and imperfect image processing or calibration result. To avoid this sensitivity, this thesis proposes a robust surface construction algorithm which initially classifies local convex regions from imperfect MC vertices and then aggregates local surfaces constructed by the 3D convex hull algorithm. Furthermore, this thesis also explores the use of wide baseline images to refine a coarse VH using an affine invariant region descriptor. This improves the quality of VH when a small number of initial views is given.

In conclusion, the proposed methods achieve a 3D model with enhanced accuracy. Also, robust surface modelling is retained when silhouette images are degraded by practical noise.

Abbreviations

2D Two Dimensional

3D Three Dimensional

AI Artificial Intelligence

B-Rep Boundary Representation

CAD Computer Aided Design

CCD Charge-Coupled Device

CDT Constrained Delaunay Triangulation

CSG Constructive Solid Geometry

DLT Direct Linear Transformation

DoF Degree of Freedom

DT Delaunay Triangulation

HD Hausdorff Distance

IR Invariant Region

LM Levenberg-Marquardt

LMS Least Mean Square

LMedS Least Median Square

MC Marching Cubes

MHD Modified Hausdorff Distance

MSER Maximum Stable Extremal Region

NN Neural Networks

pdf probability density function

PH Photo Hull

PPM Point Pattern Matching

RANSAC RANdom SAmple Consensus

SC Space Carving

SfS Shape from Silhouette

SIFT Scale Invariant Feature Transform

STL Standard Template Library

SVD Singular Value Decomposition

VC Voxel Colouring

VH Visual Hull

Chapter 1

Introduction

1.1 Introduction

Computer vision is an inter-disciplinary subject that investigates a decision-making algorithm based on the analysis of digitised images. Thus, it is highly related to image processing, pattern recognition and photogrammetry which are concerned with obtaining reliable and accurate measurements from non-contact imaging [1], and are often motivated by biological vision and psychophysics [2]. An image is a vital source of significant visual information for recognition (e.g., the colour and shape of an object), on which recent vision systems heavily rely. More specifically, these systems exploit distinctive image features, which are extracted from the visual clues and incorporated in a sophisticated decision making scenario generally devised from Artificial Intelligence (AI), Neural Network (NN) theory, and statistical analysis. Indeed, this approach retains a certain level of recognition performance.

However, it is inevitable that a feature defined on the projection of a 3D object is deprived of the information from the higher dimension, so that these clues can be sensitive to viewing conditions. For example, the shape can be distorted by viewing directions, and colour can change according to the material of an object, lighting condition and the sensor characteristic. Consequently, there are some limits to understand 3D scene solely based on 2D image features. To address this limited capability of traditional 2D features, recent researches explore a method which associates depth or volume information with 2D

features, and this thesis is also inspired by the concern regarding how to obtain accurate shape information from multiple images in a robust way.

This introduction chapter is organised as follows. Section 1.2 briefly reviews fundamental ideas of conventional vision-based 3D reconstructions, such as stereo camera, shape from silhouettes and active vision techniques. Section 1.3 shows a list of major contribution. The last section presents the outline of the thesis.

1.2 Vision-based 3D reconstruction

There are various approaches to reconstructing a 3D object or scene from images. This section briefly explains three major methods (e.g., stereo reconstruction, active vision, and shape from silhouettes) and identifies pros and cons of each approach.

1.2.1 Stereo reconstruction

Stereo reconstruction is inspired by human stereopsis, i.e., perceiving depth from binocular disparity. The realisation of this mechanism requires two cameras and the reconstruction performance is superior to that from monocular visual clues, such as texture and shading[1]. Given a pair of 2D corresponding points associated with two projection matrices, a stereo reconstruction computes a depth of the 2D point from a triangle, formed by a camera baseline and an intersection point of two back-projection rays from the pair of corresponding points. Therefore, the primary concerns in early stereo reconstruction research are: how to estimate point correspondences accurately and efficiently in practical situations, where a matching point often disappears by occlusion in another view [8, 6]; and how to minimise the triangulation error when physical lens characteristics do not comply with a linear calibration result [1, 9]. Therefore, most of early researches try to achieve robust image matching and to search for an appropriate non-linear projection model compensating a linear estimation of a projection matrix. Recently research effort has moved to self-calibration techniques which determines projection matrix only from

[1]Perceiving depth form a single view sounds fancy but it is a considerably complicated problem. For example, the shape from shading requires a solid lighting model, which is easily violated in practice. Similarly, the shape from texture also needs an initial texture model which can explain a distorted texture image [8].

image correspondences so that the reconstruction is realised without an offline camera calibration.

The shape from stereo method produces a reliable depth result. However, the result is generally given as a form of unorganised point cloud, which further requires a surface generation algorithm for realistic visualisation. The density of reconstructed points is not consistence, e.g., objects with homogeneous texture suffer lack of feature points. Furthermore, the volume of an object cannot be determined from only two views placed in front of the object. Therefore, it is unable to extract significant 3D information, such as surface curvature, surface normal vectors, and the volume of a certain region.

1.2.2 Active vision

Active vision includes all shape reconstruction methods that exploit an image of an object, on which an external energy (e.g., a laser or a light projector) is intentionally projected to alleviate the problems occurring in traditional stereo techniques. Fundamentally, it is a case that the second camera is replaced with a laser projector, so that the reconstruction is more actively achieved by means of more than visual information. As an accurate energy source, the early active vision mostly utilises a laser, though it has been replaced with a practical light projector later. For example, a line generated from a laser scans an object and images captured during scanning produce 3D information after an active triangulation, which is a modification of the traditional triangulation to exploit the scene geometry established from a projector and a camera [2]. Consequently, this line-sweeping produces sufficient number of feature points even in a textureless object and establishing point correspondences is no longer required in the reconstruction process. Furthermore, surfaces of the reconstructed points are straightforwardly constructed from the skinning algorithm [10]. To reduce the scanning time, multiple lines are often projected simultaneously in a binary coded pattern [2]. However, this approach requires to analyse a sequence of scanned images spatio-temporally, e.g., each image is processed to extract distorted line segments (an image of projected lasers) and they are classified by images captured at other time instances. As an alternative of a rapid reconstruction, all sweeping lines are encoded into a special pattern called a coded light pattern to realise the one-shot reconstruction [11].

A major disadvantage of an active system is that it needs to be equipped with a precise light projecting source, which is not cost efficient, and the object has to stay still when a line sweeps. Although a specially coded pattern can ease some constraints, it involves a pattern decoding process which is similarly complex as the correspondence problem in a stereo reconstruction. However, because of the high accuracy, it is mostly adopted in industrial applications.

1.2.3 Shape from silhouettes

Shape from Silhouette (SfS) retrieves volume information from multiple images, where the images do not need to be correlated with each other by an underlying motion, such as a sequence used in the structure from motion technique. Instead of the triangulation, the volume of an object is approximated by the intersections of cones, defined by a set of back-projection rays on an object boundary. Therefore, complicated point detection and matching processes are reduced to a silhouette image detection, i.e., it only utilises the information on whether a 2D image location is occupied by an object. The quality of the reconstructed volume relies on the viewing directions as well as the number of image used, so that it is better for multiple images to surround an object, although a rough approximation of the volume is still possible from a few shots at widely separated positions. Once the volume data is approximated, the construction of the object surface is much easier, e.g., using the marching cubes algorithm [12] to extract triangular meshes from cubic volume elements. The quality of surface meshes is adaptively controlled in recent variants of marching cubes [13].

A silhouette presents an object as binary values in an image plane, i.e., white represents a pixel occupied by an object and black is not occupied. Therefore, SfS techniques search for voxels (i.e., volume elements) not belonging to the silhouette and discard them to confine the object volume. Later developments of SfS technique exploit the colour consistency of a scene in a carving procedure. Thus, such an approach does not requires image segmentation for silhouette detection (which is not reliable in a cluttered scene), and the texture of object surface is simultaneously determined during the volume reconstruction. However, the colour consistency is only acceptable when a large number of images is involved in reconstruction, but the processing time increases exponentially with

the number is increased.

Each of three major vision-based reconstruction techniques has its own advantages and disadvantages so that the reconstruction is appropriately chosen according to its application area. Since this thesis explores a shape reconstruction as a potential 3D source of an object recognition system, a SfS method, which is solely based on images but constructs volume information without the complex matching process, is mainly explored.

1.3 Major contributions

The main contribution of this thesis is delivering a robust 3D modelling algorithm which exploits volume information obtained from multiple images of an object. To realise this goal, this thesis also investigates the following sub-research topics:

- multiple camera calibration from approximated circular motion;
- VH improvement from additional uncalibrated images;
- robust surface modelling from a volume data.

In particular, state-of-the-art image matching algorithms including image region descriptor are thoroughly explored to achieve the wide-baseline image matching, which plays a key role when tackling problems of the above topics. However, the proposed method is currently limited to static scene application with a controlled background.

1.4 Thesis organisation

This thesis contributes to the accurate 3D volume recovery from multiple views and its visualisation. To achieve this, the thesis explores five different research topics: volume reconstruction, projection matrix modelling, robust image matching, visual hull improvement and surface construction from volume data. Consequently, each chapter from Chapter 2 to Chapter 6 has been written as an independent work, which aims to the same goal. Brief explanation of each chapter are as follows.

Chapter 2 presents a detailed explanation of the volume reconstruction technique, including the underlying concept of SfS, some of the geometric entities of SfS (e.g., a

silhouette cone and frontier point), and octree representation of a volumetric data. Some pseudo codes and code snippets are presented to illustrate the silhouette detection and octree construction. In addition, this chapter analyses reconstruction results of objects with different shape complexity. Also, the latest approaches in SfS, the voxel colouring and space carving algorithms, are reviewed and compared to a traditional SfS.

Chapter 3 presents a 3D object reconstruction system which uses multiple images taken with a fixed camera of an object moving in an approximate circular motion. The circular motion is generally defined to be a pure rotation case of planar motion when modelling a turntable image sequence or equivalently an image sequence of a fixed object generated with a rotating camera. However, in practical situations the assumption of pure rotation is often violated as the rotation axis is not fixed during the rotation of a turntable. To address this problem, this chapter proposes a modified method for estimating the projection matrix associated with circular motion and an object reconstruction method based on the octree algorithm. Experimental results on several real turntable image sequences demonstrate good object reconstructions.

Chapter 4 proposes a robust image matching method. Establishing point correspondences between images is a fundamental process in many computer vision applications, e.g., motion tracking, auto calibration and object recognition. In particular, this thesis exploits it for the visual hull refinement. Before introducing the refinement algorithm, this chapter proposes two matching methods associated with Delaunay tessellation and Hausdorff distance. Delaunay tessellation describes a set of arbitrarily distributed points as unique triangular graphs which preserves most local point configuration called a clique regardless of noise addition and partial occlusion. In this chapter, this structure is utilised in a matching method and a clique-based Hausdorff Distance (HD) to address point pattern matching problems is proposed. Moreover, to enhance matching performance under affine transform, a feature descriptor, a distinctive and compact form of local information near to a feature point, is incorporated in the point pattern matching and a clique descriptor is introduced as a result.

Chapter 5 explains an algorithm that enhances the volume reconstruction from additional images. The quality of a Visual Hull (VH) depends on the number of silhouette images as well as their viewing directions. Therefore, this chapter proposes a method

which improves an initial coarse VH by using additional images captured at arbitrary positions with unknown camera parameters. The method calibrates a new image from 3D-to-2D point pairs which are obtained by epipolar transfer of two initial images and a new image, and stereo based point reconstruction.

Chapter 6 is about a robust surface construction method using volume data. The Marching Cube (MC) is a general method which can construct a surface of an object from its volumetric data generated using a shape from silhouette method. Although MC is efficient and straightforward to implement, a MC surface may have a discontinuity even though the volumetric data is continuous. This is because surface construction is more sensitive to image noise than the construction of volumetric data. To address this problem, this chapter proposes a surface construction algorithm which aggregates local surfaces constructed by the 3D convex hull algorithm. Thus, the proposed method initially classifies local convex regions from imperfect MC vertices based on sliced volumetric data. Experimental results show that continuous surfaces are obtained from imperfect silhouette images of both convex and non-convex objects. Finally, Chapter 7 summarises the thesis with some future works.

Chapter 2

Shape from Silhouettes

2.1 Introduction

Traditional scene reconstruction from stereoscopic images requires a robust feature matching process and restricts the length of a baseline of the two views (e.g., too narrow baseline degrades the accuracy of triangulation), in order to obtain an appropriate number of 3D points with good accuracy. Furthermore, in case where there are only sparse features due to the homogeneous texture of the object, it is necessary to perform parameterised surface modelling to interpolate surfaces in order to increase the density of reconstructed points. After all, the most challenging task in stereo reconstruction is to solve the feature matching problem explicitly in an occluded view, even though many algorithms have been introduced to address this issue.

However, when the surrounding views of an object and projection transforms are available, the volume of an object can be confined by the intersections of silhouette cones without point correspondences, resulting in a 3D convex hull called a Visual Hull (VH) [14]. This approach is sometimes called the volume intersection technique, but reconstruction methods involving projection (or back-projection) of multiple silhouettes are collectively referred to as the Shape from Silhouette (SfS) technique. Therefore, SfS is more robust than stereoscopic reconstructions and it can control the density of surface meshes as they depend on the predefined voxel resolutions. This chapter explains the useful geometric entities in SfS such as a frontier point, contour generator and silhouette

cone, followed by the introduction of two useful VH construction schemes. It also analyses the reconstruction performance of four objects with different shape complexity and discusses some implementing issues. Therefore, this chapter mainly explores the following topics:

- the geometric aspect of of Shape from Silhouette technique;
- two straightforward silhouette detection algorithms;
- a general implementation of octree reconstruction in addition to two bespoke algorithms developed for the thesis experiments;
- brief review of relevant researches (i.e., various volume representations) and recent variations of SfS such as voxel colouring and space carving.

2.2 Geometric entities in SfS

The basic approach of reconstruction by volume intersection is dated back to Baumgart's 1974 PhD thesis, where a polyhedral visual hull is constructed by intersecting the silhouette cones associated with the polygonal silhouettes [15]. Although the concept of VH construction is straightforward, its implementation becomes difficult without the knowledge of multi-view geometry. When an object is shown by multiple views, the boundary of the object in each image plane defines an occluding contour[1] [17], and the corresponding 3D curve that lies on the surface of an object is referred to as a contour generator, rim and limb [18]. Thus, an occluding contour is the result of the projection of a contour generator. In other words, the back-projection of a point on an occluding contour creates a ray, a half-infinite vector that starts from a camera centre, and a set of all rays of boundary points constructs a cone in the 3D space, called a silhouette cone or a viewing cone.

This relationship is illustrated in Figure 2.1, where the world, $(i-1)$, i and $(i+1)$-th camera frames are respectively represented by their origins with respect to the world frame, i.e., $\vec{o}^{w}$, $\vec{c}^{w}_{(i-1)}$, $\vec{c}^{w}_{i}$ and $\vec{c}^{w}_{(i+1)}$. The three bold arrows near the origin represent the three basis vectors of each frame, and image planes, I_{i-1}, I_i and I_{i+1} invert their physical

[1] It is also called a profile [16] or an apparent contour [15] by other authors.

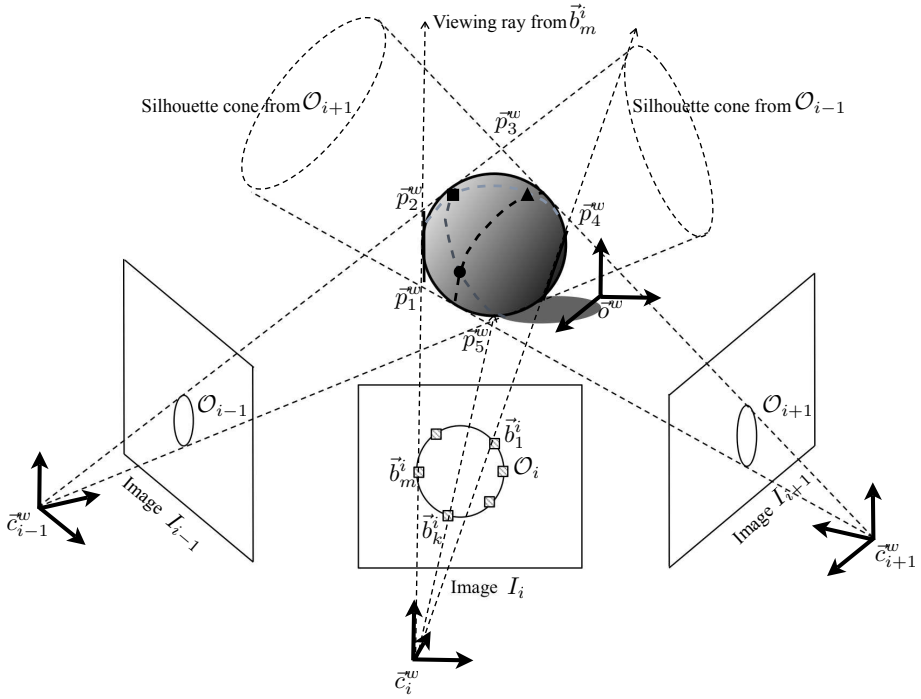


Figure 2.1: Shape from Silhouette approach: Three views with camera centre $\vec{c}_i^w$, $\vec{c}_{(i-1)}^w$ and $\vec{c}_{(i+1)}^w$ produce silhouette cones and intersections (e.g., $\vec{p}_1^w$, $\vec{p}_2^w$, $\vec{p}_3^w$, $\vec{p}_4^w$, and $\vec{p}_5^w$) approximate the volume of an object. Dotted lines on the object denote each contour generator and frontier points are marked as triangle, circle and square.

positions in front of the camera centres for illustration purpose. In the image plane I_i, the occluding contour $\mathcal{O}_i$ is extracted from the boundary of an object silhouette, i.e.,

$$\mathcal{O}_i = \{\vec{b}_1^i, \vec{b}_2^i, \cdots, \vec{b}_m^i \ | d_{\mathrm{E}}(\vec{b}_k^i, \vec{b}_{k+1}^i) \leq \sqrt{2},\ 0 \leq k < m\}, \tag{2.1}$$

where $d_{\mathrm{E}}(\cdot)$ denotes Euclidean distance of two points and $\vec{b}^i$ is a point on the external boundary of the silhouette, so that $\mathcal{O}_i$ defines a closed contour from a set of 8-connected neighbour points. A silhouette cone whose apex is a camera centre, is then constructed from a set of rays as illustrated by dotted lines in Figure 2.1 and points on an object that touches a silhouette cone define a contour generator (see dotted curves on the object). The intersection of three silhouette cones confines the space occupied by an object and

this convex region becomes a VH of an object, i.e,

$$\mathrm{VH} = \bigcap_{i=1} cone(\mathcal{O}_i, P_i), \tag{2.2}$$

where $cone(\cdot)$ is a function that constructs a silhouette cone by back-projecting an occluding contour using a projection matrix P_i. Thus, a polygonal region defined by $\vec{p}_1^{\,w}$, $\vec{p}_2^{\,w}$, $\vec{p}_3^{\,w}$, $\vec{p}_4^{\,w}$ and $\vec{p}_5^{\,w}$ approximate the sphere shown in Figure 2.1. However, self-occluded parts (e.g., small components behind the sphere) cannot be reconstructed correctly in this view configuration and some concave regions which cannot be differentiable in the image plane (e.g, inside of a mug) are reconstructed as convex.

Thus, one of the major concerns in early SfS is how to efficiently estimate the intersection points of visual cones, and approaches regarding this problem are broadly divided into two groups in terms of the space in which the intersection test takes place. For example, with an approach of the first group, a 3D object is reconstructed by back-projecting its 2D silhouettes (generated from different views) onto the 3D space, i.e., intersection test is performed in a 3D space. On the contrary, an approach belonging to the second group projects approximately known 3D object positions onto the images and carves the shape if the projections lie outside the corresponding silhouettes, i.e., the intersection test is performed in a 2D space. Since the complexity exponentially increases as the dimension of space increases, the approaches in the latter group are normally preferred in recent SfS techniques. The other advantage of the 2D intersection test is that it can integrate an octree structure easily into SfS for organising voxels.

Another important geometric entity is defined at the intersections of contour generators. In general, it is difficult to establish point correspondences solely based on two occluding contours, because the shape of a silhouette is dependent on the viewing direction. However, intersections of contour generators can give significant clue for the boundary matching, and they are called frontier points, denoted by •, ▲ and ■ in Figure 2.1. The methods shown in [15, 19] proposes VH construction by means of this characteristic of frontier points.

2.3 Visual hull representation

2.3.1 Octree representation

In computer graphics, surface models are generally represented by lists of vertices, edges and faces, which are efficient in terms of rendering complex surface model, but it is not the best representation for describing the volume of an object. Instead, the general volume representation utilises various types of solid primitives (e.g., polyhedron, cylinder and sphere) to describe an object by the combinations of these primitives. This volume representation was widely utilised in early Computer Aided Design (CAD), which required a suitable 3D model for the low level of graphical display [10]. As a combination method of volume primitives, the Boundary Representation (B-Rep) describes the volume by surface patches and their connection graph. The other method called Constructive Solid Geometry (CSG) relates volume primitives using a tree structure of operation [20]. However, B-Rep and CSG has limits when modelling arbitrary shape of an object. Furthermore, it is not easily incorporated in SfS techniques.

An octree is a hierarchical tree structure of voxels and one octree is sufficient to represent the volume of an object. Each node of an octree can have eight child nodes, and the size of each dimension of a child node is half of the size of its parent node. Therefore, the volume resolution of a VH becomes two times finer whenever the current node produces offspring. To incorporate this structure into SfS, an initial bounding voxel, i.e., an octant, should have sufficient volume to include an object, and an octant then produces child nodes (i.e., split into 8 smaller octants) if a current octant intersects (or includes) an object until a tree meets the predefined level. For example, Figure 2.2(b) illustrates a three-level octree formed from a simple object shaded as grey in Figure 2.2(a), where dotted cubes indicate nodes of an octree. The indices of the second level of an octree are shown in both figures (a) and (b) for a direct comparison of a node in the tree with its corresponding position in the 3D space.

Every node in an octree has a status, i.e., intersection, inside or background, which indicates the result of an intersection test. The white, grey, and black rectangles in Figure 2.2(b) respectively denote the background, intersection, and inside octants. Thus, final reconstruction is accomplished by removing all octants with background status.

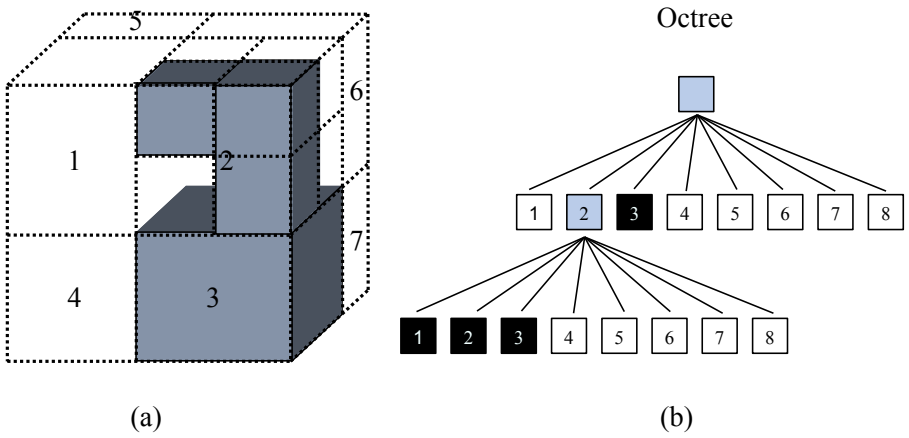


Figure 2.2: Example of an octree: (a) Grey voxels indicates the actual shape of an object, whilst dotted cubes represent the shape of octants. The indices correspond to the node indices in the second level of octree in (b); (b) A three-level of an octree of the object shown in (a), where black, white and grey node represent inside, background and intersection octant, respectively.

Classified as intersection, the octant is split into eight sub-octants to describe the object in more details until the level of an octree reaches the predefined level. An initial voxel enclosing an object is pre-determined manually, i.e., the root octant is the largest voxel that consists of eight child octants indexed from 1 to 8 in Figure 2.2(a), where the eighth octants is occluded by the fourth octant. Since the initial octant in Figure 2.2(a) includes a grey object (i.e., an intersection case), it is split and the intersection test is repeatedly applied to these smaller octants. Since the second octant in the second level of an octree is classified as an intersection octant while the third octant is an inside octant, only the second node is split into eight child nodes. Thus, four octants describes the object, i.e., the first, second, and third octants in the last level of the octree, and the third octant in the second level of the octree.

As explained in Section 2.2, the intersection test can be performed faster in the image plane. Thus, eight corner points of each octant are projected onto every image plane by a camera projection matrix. The projections of corners generate a six-sided polygon in the image plane, but the shape is normally approximated by a rectangle or a square before the intersection test to further expedite the process. An algorithm proposed in [21] focusses on the fast decision-making strategy using approximation of a projected cube.

Table 2.1: Octree updating rule.

Previous \ Current	**Unknown**	**Background**	**Intersection**	**Inside**
Unknown	Unknown	Background	Intersection	Inside
Background	Background	Background	Background	Background
Intersection	Intersection	Background	Intersection	Intersection
Inside	Inside	Background	Intersection	Inside

For example, each projected cube is converted into a bounding square and the status of the bounding square is determined by single lookup of two distance maps constructed from silhouette and its complement [21].

When an additional image is supplied, it is used to update the previous status of an octant by constructing a new silhouette cone. The updating rule is summarised in Table 2.1, where the status of a current octant estimated by a new silhouette image is shown in the first row of the table and the previous statuses are presented in the first column. As shown, the background status has top priority, i.e., it overrides other statuses. The priority of other statuses is ordered as intersection $>$ inside $>$ unknown, which is only required when describing the status of an initial octant. Alternatively, an octree can also be constructed by the quadtrees of silhouettes, which traverses a tree structure more efficiently than the traditional octree algorithm. However, three quadtrees from the orthogonal directions are required to represent one object. Also, the projection method is restricted as orthogonal transform [22], which is only satisfied when a camera is placed in a distance from an object.

2.3.2 Silhouette detection

The SfS requires accurate silhouettes of an object for the intersection test, but it is a challenging task to devise an universal algorithm which adaptively processes an image for a silhouette detection regardless of image conditions. Although there are some methods to extract an object boundary from cluttered scene using an active contour [23], most algorithms belonging to this category are based on iterative computation from many manually selected thresholds with a good initial guess. Also, the result is not reliable in the presence of large amount of clutter in the background, non-homogeneous texture of an object and shading due to a directional lighting. Nonetheless, the silhouette detection is regarded as

a preprocessing procedure in the whole reconstruction system, so that it needs to be computationally simple (e.g., using intensity thresholding) but should retain a certain level of robustness. Assuming a controlled scene (i.e., an object to be reconstructed is placed in a homogeneous background under controlled lighting condition), some fundamental image processing techniques such as Gaussian blurring, contrast enhancement and histogram equalising, are selectively applied to an input image to improve the performance of the silhouette detection by intensity thresholding.

However, this approach may include an artefact inside an object because of the object texture having similar colour to background or shading produced by self-occlusion. To address this problem, an occluding contour filled with the maximum intensity values is used to represent a silhouette. Suppose that I_i' is a thresholded image from the i-th image I_i, i.e.,

$$I_i'(\vec{p}_k^{\,i}) = \begin{cases} 1 & \text{if } I_i(\vec{p}_k^{\,i}) \geq \tau_{int} \\ 0 & \text{otherwise} \end{cases}, \tag{2.3}$$

where τ_{int} denotes an intensity threshold. In practice, I_i' may contain more than one segment, which is the aggregation of connected nonzero points, even though I_i is grabbed from a controlled scene and enhanced with some image processing techniques. Assuming there is only one object in I_i, an occluding contour of an object is selected by a set of connected boundary points with the maximum length in the image plane, i.e.,

$$\mathcal{O}_i = \arg\max_k(|\mathcal{O}_k|), \tag{2.4}$$

where $|\cdot|$ indicates the cardinality of a set. Thus, a silhouette is produced by filling inside the closed boundary set $\mathcal{O}_i$ to produce a silhouette $\mathcal{S}_i$ in the i-th image. If inside cavities of an object (e.g., handle of a mug) are taken into consideration in the reconstruction, the internal boundaries of the largest segment are also utilised to accomplish the silhouette detection, i.e., a set of internal boundaries is

$$\mathcal{O}_{\text{int}} = \{\bigcup_k \mathcal{O}_k \ | \mathcal{O}_k \in \mathcal{S}_i\}. \tag{2.5}$$

Thus, after an external contour with the maximum length amongst all detected contours

```
1  IplImage*CImgProc::
   GetSizeFilterImg(IplImage* imgBW, int nIntCon)
3  {
       ////////////////////////////////////////
5      // initialise a return buffer
       ////////////////////////////////////////
7      IplImage* imgRes = NULL;
       imgRes = cvCreateImage(cvGetSize(imgBW), 8, 1);
9      if (imgBW->nChannels > 1) return imgRes;

11     ////////////////////////////////////////
       // create memory storage
13     ////////////////////////////////////////
       CvMemStorage* storage = cvCreateMemStorage(0);
15     CvSeq* contour = NULL;
       CvContourScanner scanner;
17     scanner = cvStartFindContours(imgBW, storage,sizeof(CvContour), CV_RETR_CCOMP,
           CV_CHAIN_APPROX_NONE);
       contour = cvFindNextContour( scanner );
19
       ////////////////////////////////////////
21     // define contour structure
       ////////////////////////////////////////
23     struct SContours
       {
25         vector<CvPoint>* parContour;
           int nTotCon ;
27         int nLongestConIdx ;
           int nMaxLength;
29     };

31     ////////////////////////////////////////
       // initialising
33     ////////////////////////////////////////
       SContours allContours;
35     allContours.nTotCon = 0;
       allContours.nLongestConIdx = -1;
37     allContours.nMaxLength = -1;
```

Figure 2.3: Size filtering code snippet 1.

in the image plane is selected as an occluding contour, the seed-flooding algorithm [24] follows to fill the inside of the contour to construct an initial silhouette. If internal holes are further needed, internal contours are overlaid to the previously obtained silhouette, and filled with the lowest intensity values. Since this method requires to sort all contours in terms of their length, it is possible to filter (i.e., remove) small contours and selectively include internal contours. Thus, this silhouette detection and filtering process is called a size filtering in this thesis.

An image processing class CImgProc is designed to accomplish this goal. It includes algorithms from simple image processing functions to the size filtering, which have been developed mostly based on the Intel OpenCV library [25]. For example, the function GetSizeFilterImg(·), as shown in Figure 2.3, is called with two input parameters, e.g., a pointer of an initial thresholded image and the number of internal contours required. Thus, if a variable nIntCon in line 1 is set to zero, the function returns a silhouette image estimated only from an external object boundary. Lines from 7 to 9 create a memory buffer that stores the resulting silhouette image, initialised as a 8-bit grey image with the same size of the input image, imgBW. CvMemStorage in line 14, is a OpenCV data type

```
1  ////////////////////////////////////////
   // allowing inside cavities
3  ////////////////////////////////////////
   if (nIntCon > 0)
5  {
       ////////////////////////////////////////
7      // 1. hit test and remove the contours outside an object
       ////////////////////////////////////////////////
9      vector<int> arInsideContourIdx;
       for (int i = 0 ; i < allContours.nTotCon; i++ )
11     {
           if (i == allContours.nLongestConIdx) continue;
13         bool bIsInside = true;
           for (int j = 0 ; j < allContours.parContour[i].size() ; j++)
15         {
               CvPoint2D32f pt;
17             pt.x = allContours.parContour[i].at(j).x;
               pt.y = allContours.parContour[i].at(j).y;
19             if (cvPointPolygonTest(contour, pt,0) < 0)
               {
21                 bIsInside = false;
                   break;
23             }
           }
25         if (bIsInside)
               arInsideContourIdx.push_back(i);
27     }

29     ////////////////////////////////////////
       // 2. ordering
31     ////////////////////////////////////////
       for(int i = 0; i < arInsideContourIdx.size(); i++)
33     {
           for(int j = 1 ; j < arInsideContourIdx.size() - i ; j++)
35         {
               if ( int (allContours.parContour[ int (arInsideContourIdx[j]) ].size()) >
37                  int (allContours.parContour[ int(arInsideContourIdx[j-1]) ].size()) )
               {
39                 int nTemp;
                   nTemp = arInsideContourIdx[j];
41                 arInsideContourIdx[j] = arInsideContourIdx[(j-1)];
                   arInsideContourIdx[(j-1)] = nTemp;
43             }
           }
45     }

47     ////////////////////////////////////////
       // 3. draw black internal contours
49     ////////////////////////////////////////
       if (nIntCon > arInsideContourIdx.size())
51         nIntCon = arInsideContourIdx.size();

53     for(int i = 0; i < nIntCon; i++)
       {
55         cvEndFindContours( &scanner );
           scanner = cvStartFindContours(imgBW, storage,sizeof(CvContour), CV_RETR_CCOMP,
                CV_CHAIN_APPROX_NONE);
57         contour = cvFindNextContour( scanner );
           for (int j = 0 ; j < arInsideContourIdx[i]; j++)
59             contour = cvFindNextContour( scanner );
           CvScalar color = CV_RGB(0,0,0);
61         cvDrawContours(imgRes, contour, color, color, 1, CV_FILLED, 8);
       }
63 }
```

Figure 2.4: Size filtering code snippet 2.

that creates a stack memory structure to enclose a dynamic memory buffer (e.g., a linked list).

The contour searching process in line 18 is provided by a function of OpenCV (i.e., the process is optimised to an Intel processor[2]) and it returns the boundary of an image segment as a pointer to a linked list, CvSeq*. Before calling the function cvFindNextContour(·), it is required to initialise the contour retrieving properties (see line 17). One can directly use a CvSeq pointer whenever it is needed, but all contour

[2]OpenCV is optionally optimised highly by loading the commercial Intel Integrated Performance Primitives (IPP) [26], which provides high performance low level routines.

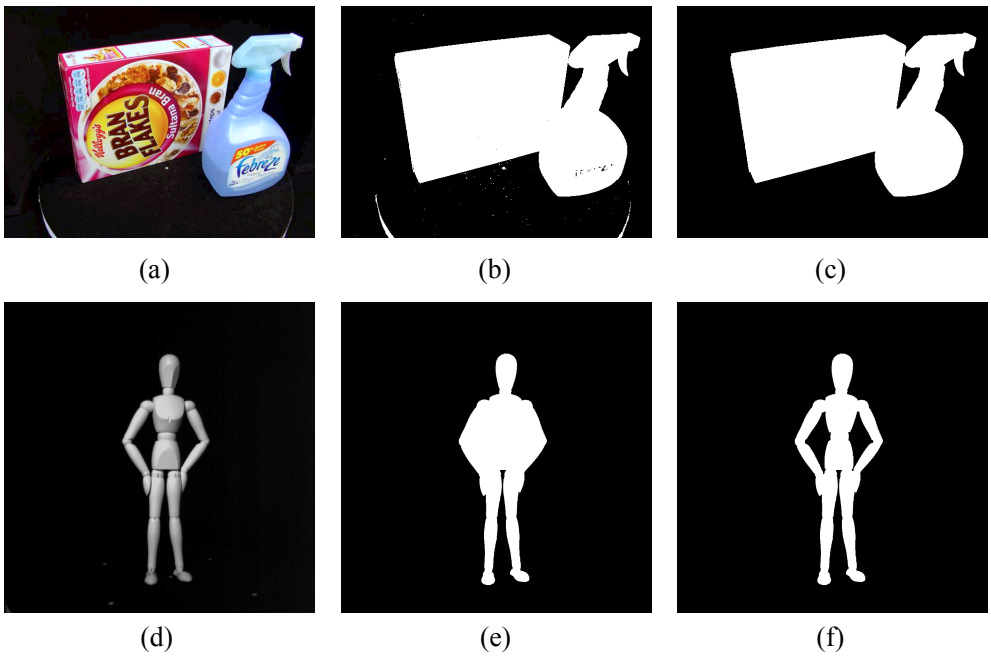


Figure 2.5: Example of silhouette detection: (a) objects with non-homogeneous texture; (b) thresholding result of (a); (c) a silhouette is estimated by applying the flood-seeding algorithm to the largest occluding contour; (d) an object with homogeneous texture but two internal cavities; (e) two inside holes cannot be distinguishable after the food-seeding algorithm; (f) two internal segments complete the silhouette detection.

data are temporarily organised in a user-defined local structure SContours (see lines 23 to 29) to facilitate data access. Thus, a variable, allContours defined in line 34, includes a pointer of a vector data type [3] for storing all contours. Additionally, other useful variables are temporarily stored, e.g., the longest contour index (i.e., an occluding contour), the total number of segments in an image plane, and the length of the occluding contour.

Internal contours are used to create inside holes on an initial silhouette as many as the input parameter (i.e., nIntCon) indicates. This process comprises of three steps: searching internal contours, ordering internal contours with respect to their length, and painting the inside of a hole with the lowest intensity values. More details of these processes are shown in Figure 2.4, which is a part of the function GetSizeFilterImg(·) in Figure 2.3. A variable arInsideContourIdx in line 9 stores all indices of internal contours, determined by a hit test shown from lines 9 to 27. The bubble sorting estimates the influence of an internal contour, i.e., the longer contour is the more influential it is. The

[3] It is found in a Standard Template Library (STL) of ANSI C++.

worst case complexity of the sorting algorithm is $O(n^2)$ [27], but the processing time in practical applications is acceptable because the number of segments in an image is not significantly large in a controlled scene. Finally, a function cvDrawContours(·) in line 61 paints inside of internal contours.

Figure 2.5 shows some results of the silhouette detection using the size filtering algorithm. When an image does not have homogeneous texture, it is likely to have small inside artefacts after thresholding. For example, an image shown in Figure 2.5(a) has two objects (e.g., a cornflake box and a spray), and there are low intensity regions in the object area (e.g., the label of the spray). In addition, Salt and pepper noise may appear in a colour image sensor without Gaussian blurring [see the white dots in the turntable area in Figure 2.5(b)]. Also, some bright regions outside an object can be presented in an initial thresholded image [see the rim of the turntable at the bottom of Figure 2.5(b)]. The largest occluding contour with white filling excludes them in a silhouette image. As a result, the algorithm generates a clean silhouettes as shown in Figure 2.5(c). On the other hand, when an object has homogeneous texture but includes inside cavities, internal boundaries are required to complete the silhouette detection. In Figure 2.5(d) the two arms of a dummy produce two holes, which disappear if the largest occluding contour is only used [see Figure 2.5(e)]. However, more details are retained after allowing two internal contours as shown in Figure 2.5(f).

2.3.3 Octree construction

Two classes, one for describing a node of an octree named COctant and the other for establishing an actual tree using COctant variables, are key classes for the C++ implementation of octree construction. Private member variables of an octant class are listed in Figure 2.6, where four integer values from -1 to 2 are also assigned to constant variables[4] representing intersection results (line 4-7). Thus, a member variable m_nStatus, which indicates the status of an octant, only have one of these constants. To enhance code reusability, the class is designed to also include colour information associated with the corners of an octant (see line 26) in addition to their 3D corner positions (i.e., 24 double buffers are assigned for colour information of eight vertices in line 24). Also, it

[4]The 'inside' status is re-named as OBJECT in COctant class declaration.

```
1  ////////////////////////////////////////
   // status constant
3  ////////////////////////////////////////
   #define BACKGROUND 0
5  #define OBJECT 1
   #define INTERSECTION 2
7  #define UNKNOWN -1

9  #include <OpenCV/OpenCV.h>

11 class COctant
   {
13 public:
       ////////////////////////////////////////
15     // constructors and destructor &
       // other interface functions
17     ////////////////////////////////////////
       ...
19
   protected:
21     ////////////////////////////////////////
       // member variables
23     ////////////////////////////////////////
       CvPoint3D32f    m_pts3DVertex[8];
25     //each scalar has colour in BGR order and the last value is vote
       CvScalar        m_scColour[8];
27     float           m_fSizeOfCube;
       int             m_nStatus;
29     COctant*        m_poctParent;
       COctant*        m_poctChildren[8];
31     bool            m_bNeedToDeleteIt;
   }
```

Figure 2.6: COctant declaration snippet 3

includes memory addresses of parent and children octants in m_poctParent (line 29) and m_poctChildren (line 30), respectively, which enable a octree class to traverse nodes. A boolean variable defined in line 31 indicates whether this octant needs to be destroyed in an octree in accordance with the background status after a status update.

COctant is extensively utilised in COctree, which defines functions that arrange node memory and update tree information. The class stores an octree in a matrix with variable column length for the memory efficiency, i.e., the $i-$th row vector in the octree matrix represents the $i-$th level of octree and the number of column elements is dependent on the population in the generation. An octree is a similar to a family tree. For example, octants in the same level of octree forms a generation, so that nodes in the row vector of the octree matrix are also called siblings. If a node is deceased, all descendants produced from the node no longer exist, i.e., a node cannot be inserted in the middle of a tree without an ancestor. One restriction on an octree is that a node always produces eight children. Thus, the maximum population of the i-th generation is $8^{(i-1)}$, which means in the worst case m generation of an octree requires $\sum_{i=1}^{m} 8^{(i-1)}$ octants, and this exponential growth of nodes makes an octree implementation slow and memory-intensive process. Therefore, when the rectangular memory structure is used, it is easy to implement but only few cells are used in the early generation. To reduce this memory redundancy, the octree matrix

```
1. create COctree instance
   // at this stage there is no children but only the family founder exists
2. make full family member in the last generation, i.e., 8^{(j-1)} octants

for i = 0 until the end of images
   for k = 0 until end of octants in the last generation
      3. project the k-th octant onto the i-th image
      4. decide the status of an octant
         // i.e., the complex status updating rule is unnecessary
   end for
   erase all background octants
end for
```

Figure 2.7: Octree construction method I snippet 4.

is designed to allow the variable length of a row vector, which is realised by adopting a liked list structure to store siblings. For example, an array of pointers of sub-octants in the same generation is stored as a linked list of COctant*, and multiple arrays from all levels of the tree forms the octree matrix.

Two algorithms are implemented for the construction of an octree incorporating the intersection test. A pseudo code shown in Figure 2.7 is designed to only exploit octants in the last generation. If the last level of an octree is known, it is able to estimate the total number of octants in the worst case. Therefore, two iterations [i.e., per view iteration (see line 5 in Figure 2.7) and per octant iteration in the last generation (see line 6) in Figure 2.7] are sufficient to construct an octree, and the worst case complexity of the algorithm is $i \times \exp(j)$ (i.e., $i \times 8^j$), where i and j represent the number of silhouettes and the last level of an octree. It is an advantage of this method that iterations for the progressive tree construction is not involved, i.e., functions for making offspring or destroying descendants are not called inside the iteration. However, the initial octree construction from the first two views takes most of construction time if the level of an octree is high because all octants in the last generations are unnecessarily projected for the intersection test. Furthermore, the resolution of a simple shape is unnecessarily high because the a VH only uses the finest resolution of an octant, which consequently increases the size of final 3D data.

The second algorithm shown in Figure 2.8 is more memory efficient and faster than method I, if the shape of an object is not significantly complex. The octree construction method II consists of three major processes: initialising an octree (line 1), projecting an octant (line 7), and updating status of a node (line 9), which are incorporated in triple iterations. The worst case complexity of the algorithm is $i \times \exp(j!)$ (i.e., $i \times 8^0 \times \cdots \times 8^j$),

```
1.create COctree instance
// at this stage there is no child in a tree but only family fonder exists

for i = 0 until the end of image
   for j = 0 until max_Octree_gen
      for k = 0 until the end of octants in j-th generation
         2. project the k-th octant onto the i-th image plane
         3. update the status of the k-th octant by a function
            UpdateStatus(curStatus, newStatus)
      end for
   end for
end for
```

Figure 2.8: Octree construction method II snippet 5.

which is higher than the worst case complexity of the method I. However, the algorithm does not need to project all octants of the last generation in the initial reconstruction because background octants are removed in the early generation. Also, the peak memory usage drops in method II, even though it stores intermediate reconstruction results. This is because only less than 5 per cent of the maximum population is utilised to describe a VH in practice. Therefore, the total number of projections required in method II is less than that for method I, particularly when the level of octree is higher. However, when an object has complex shape, method II suffers from extensive status updating, which involves excessive function calls for the children creation or destruction of descendants. Therefore, a function UpdateStatus(·) shown in line 9 becomes a bottleneck of the computation.

2D intersection of an octant with a silhouette is generalised as three cases. The first case is that the projection of one of the corner points exists inside an object. The second case is that one of edges of the projections is across a silhouette. The last case is that a silhouette is inside a projection, but no edge and corner touch a silhouette. An algorithm shown in Figure 2.9, determines a status on a coarse-to-fine basis to minimise computation time. Suppose that a 4×8 matrix N stores 3D points of eight vertices of an octant, i.e.,

$$N = \begin{bmatrix} x & x+w & x+w & x & x & x+w & x+w & x \\ y & y & y+w & y+w & y & y & y+w & y+w \\ z & z & z & z & z+w & z+w & z+w & z+w \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \tag{2.6}$$

where w is a width of a cube. To define the status of an octree node N, a cost function

of a 3D point is firstly defined as

$$cost_{\mathrm{p}}(\vec{x}^{w}, P_i) = \begin{cases} 1 & \text{if } \vec{x}^{i}_{j} = P_i \vec{x}^{w},\ \vec{x}^{i}_{j} \in \mathcal{S}_i \\ 0 & \text{otherwise} \end{cases}, \tag{2.7}$$

where P_i denotes the i-th projection matrix. Therefore, when m images capture the same object, the status of an octant is classified as the inside octant, if it satisfies a sufficient condition, i.e.,

$$\frac{1}{8m} \sum_{i=1}^{8} \sum_{j=1}^{m} cost_{\mathrm{p}}(\vec{n}_i, P_j) = 1. \tag{2.8}$$

A sufficient condition of the background octant is

$$\prod_{i=1}^{8} \prod_{j=1}^{m} cost_{\mathrm{p}}(\vec{n}_i, P_j) = 0. \tag{2.9}$$

Thus, one sufficient condition of the intersection octant is an octant does not satisfy with (2.9) and $0 < (2.8) < 1$. To complete the condition for the intersection octant, (2.7) should be modified to cope with the edge intersection case using the condition in (B.3) and (B.4), i.e.,

$$cost_{\mathrm{e}}(\vec{x}^{w}_{i}, \vec{x}^{w}_{j}, P_k) = \begin{cases} 1 & \text{if } (\vec{x}^{i}_{l})^{\mathrm{T}} [P_k \vec{x}^{w}_{i}]_{\times} [P_k \vec{x}^{w}_{j}] = 0, \forall \vec{x}^{i}_{l} \in \mathcal{S}_i \\ 0 & \text{otherwise} \end{cases}. \tag{2.10}$$

Therefore, the second sufficient condition of the intersection octant is

$$\prod_{k=1}^{m} cost_{\mathrm{e}}(\vec{n}_i, \vec{n}_k, P_k) = 0, \tag{2.11}$$

where $d_{\mathrm{E}}(\vec{n}_i, \vec{n}_k) = w$, and the last sufficient condition of the intersection octant is defined when a convex hull of N includes $\mathcal{S}_i$.

For example, a function DecideStatus($\cdot$) shown in Figure 2.9 is designed to return a status of an octant using three input parameters: a silhouette image (imgBin), a projection matrix (matCali) and eight corner points (*ppts3DVer). The projection of a 3D point is estimated from line 15 to 31, and a variable nCount, which counts the

inside projection, is increased unless the projection is outside an object (see line from 36 to 43). Therefore, when nCount is greater than 0 but not 8, an octant is classified as intersection, while a silhouette is considered as an inside octant if nCount is 8. In practice, most intersections belong to these cases. However, when all corner points are outside a silhouette, a refined intersection test is performed in a function TestInclusion(·), which approximates the projections as 2D rectangle and check if there are any nonzero values in the same rectangle region of imgBin, i.e., this function checks the second and third intersection case.

```
DecideStatus(IplImage* imgBin, CvMat* matCali, CvPoint3D32f* ppts3DVer)
{
    int nRes = UNKNOWN;
    int nCount = 0 ;
    CvMat *mat2D[8], *mat3D[8];

    for(int i = 0 ; i < 8 ; i++)
    {
        mat2D[i] = cvCreateMat(3, 1, CV_32F);
        mat3D[i] = cvCreateMat(4, 1, CV_32F);

        ////////////////////////////////////////
        // 3d data
        ////////////////////////////////////////
        cvmSet(mat3D[i], 0 , 0, ppts3DVer[i].x);
        cvmSet(mat3D[i], 1 , 0, ppts3DVer[i].y);
        cvmSet(mat3D[i], 2 , 0, ppts3DVer[i].z);
        cvmSet(mat3D[i], 3 , 0, 1.0);

        ////////////////////////////////////////
        // project 8 points
        ////////////////////////////////////////
        cvMatMul(matCali, mat3D[i], mat2D[i]);

        ////////////////////////////////////////
        // normalise
        ////////////////////////////////////////
        cvmSet(mat2D[i], 0, 0, cvGet2D(mat2D[i],0,0).val[0] / cvGet2D(mat2D[i],2,0).val[0]);
        cvmSet(mat2D[i], 1, 0, cvGet2D(mat2D[i],1,0).val[0] / cvGet2D(mat2D[i],2,0).val[0]);
        cvmSet(mat2D[i], 2, 0, 1.);

        ////////////////////////////////////////
        // count the number of inside corner points
        ////////////////////////////////////////
        if (cvGet2D(mat2D[i],0,0).val[0] < 0 || cvGet2D(mat2D[i],0,0).val[0] > imgBin->width ||
            cvGet2D(mat2D[i],1,0).val[0] < 0 || cvGet2D(mat2D[i],1,0).val[0] > imgBin->height)
            continue;
        else
        {
            if ( cvGet2D(imgBin, cvGet2D(mat2D[i],1,0).val[0],cvGet2D(mat2D[i],0,0).val[0]).val
                [0] > 0)
                nCount++;
        }
    }

    ////////////////////////////////////////
    // decide a status by corner position
    ////////////////////////////////////////
    if (nCount > 0 && nCount < 8) nRes = INTERSECTION;
    else if (nCount == 8) nRes = OBJECT;
    else      // i.e., nCount == 0
    {
        if (TestInclusion(mat2D,imgBin)) nRes = INTERSECTION;
        else  nRes = BACKGROUND;
    }

    ////////////////////////////////////////
    // remove buffer
    ////////////////////////////////////////
    for (int i = 0; i < 8 ; i++)
    {
        cvReleaseMat(&mat2D[i]);
        cvReleaseMat(&mat3D[i]);
    }

    return nRes;
}
```

Figure 2.9: Intersection test snippet 6.

2.3.4 Status decision in 3D space

As shown in Section 2.3.3, the most time consuming procedure in the general octree construction algorithm is the intersection test, which estimates the status of an octant. Although [28] states that projecting a voxel onto each image is more efficient than back-projecting each silhouette to an initial octant, some novel methods address this inefficiency in a 3D space. One reason which makes the 3D intersection complex, is that it is not straightforward to describe mathematically the shape of the back-projection of a silhouette. To avoid this, a silhouette is initially approximated as a convex polygon, which is further decomposed into a set of convex components. As a result, the intersection performs more efficiently with pyramids, constructed from the back-projection of convex components, than non-convex silhouette cones, because the explicit representations of convex components in a pyramid [29, 30].

Furthermore, the pyramidal intersection test in [29] does not require an updating rule for an otree between images, because the algorithm is designed to create an independently perfect octree per an image. Therefore, the last octree is simply obtained by unifying all octrees, which enables parallel computation for the intersection test. Sarivastava et al. also proposed an algorithm, which estimates a status of octant in a 3D space by means of a coarse-to-fine intersection test strategy (i.e., it rules out an inside octant in earlier stage) [30]. For example, a polygonal pyramid and an octant with unknown status are initially approximated as the smallest enclosing cone and the smallest enclosing sphere, respectively.

2.3.5 Experimental results

In this section, six experiments are performed to analyse the performance of the SfS-based volume reconstruction technique. These tests include:

- SfS volume reconstruction of four different objects with the octree representation. This test demonstrates the visual quality of a SfS result relative to the shape of an object;
- Comparing the total number of octants relative to the level of an octree. This shows how many octants are generally created as the level increases;

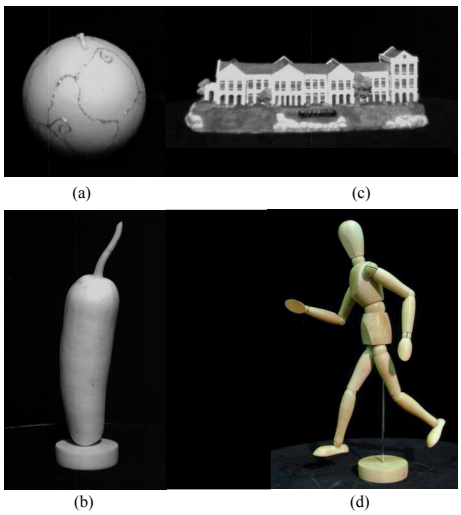


Figure 2.10: Four objects with different shape complexity.

- Comparing inside and intersection volume with respect to the level of octree. This test helps to measure the accuracy of the approximation;
- Evaluating processing time of two proposed octree construction methods;
- Intermediate reconstruction results in an octree. This test demonstrates visual quality of approximation relative to the level of octree;
- VH results relative to viewing directions.

The performance of two construction methods are evaluated using four objects with different shape complexity, such as a ball-shape candle, a model of a school, a courgette, and a dummy with running motion, which are presented in Figure 2.10(a)-(d) and ordered by the shape complexity, e.g., (a) represents the simplest convex shape; there are relatively more concave regions are involved in the object shown in (d). Sixty images of each of the objects are generated during the rotation of a turntable, on which an object is placed at a fixed camera position.

Four VH's, constructed with seven-level of octrees from these objects, are illustrated in Figure 2.11, where each result is visualised in terms of three representations, e.g., a point view, a wireframe view, and a face view (i.e., hidden points are removed). In particular, to emphasise the position and quantity of inside octants, they are highlighted in red in the first two columns. Since more details can be retained when an initial octant

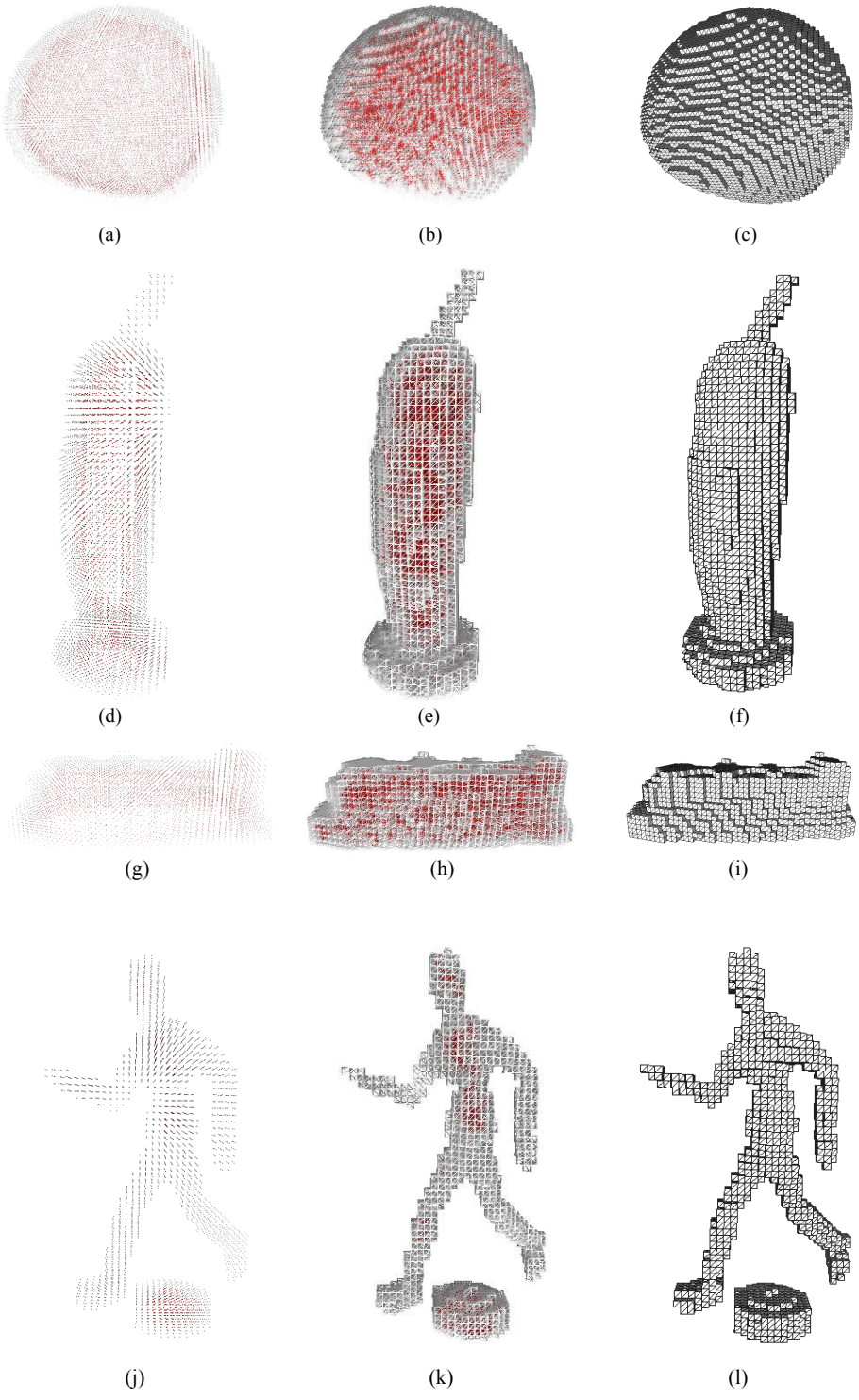


Figure 2.11: Reconstruction results: (a)-(c) reconstruction results of the object shown in Figure 2.10(a); (d)-(f) reconstruction results of the object in Figure 2.10(b); (g)-(i) reconstruction results of the object in Figure 2.10(c); (j)-(l) reconstruction results of the object in Figure 2.10(d); Each reconstruction is presented as point view, wireframe view and face view in three columns, and internal octants are highlighted in red.

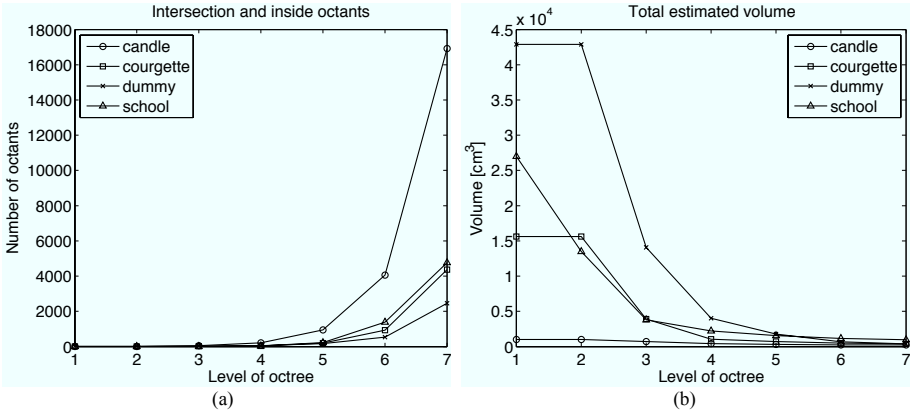


Figure 2.12: Total octants and volume of four objects: (a) total number of octant; (b) its volume relative to the level of an octree.

tightly encloses an object, the test uses four different sizes of an initial octant[5], i.e., $w =$ 10, 25, 30 and 35[cm], resulting in respectively 16920, 4368, 4766 and 2463 octants for object (a), (b), (c) and (d) in Figure 2.10. Therefore, the finest resolutions of four VH's are 0.039, 0.098, 0.117 and 0.138[cm], respectively.

The total octants (i.e., the sum of internal and inside octants) of four objects relative to the level of an octree, are shown in Figure 2.12(a), and their volumes are presented in Figure 2.12(b). The volumes of a candle and a courgette, do not decrease significantly in the second level of an octree, but they only increase the number of octants. This occurs either when an initial octant is too large or when an object is widely spread out in the initial octree. As explained in Section 2.3.3, the number of octants are exponentially increased in all cases [see Figure 2.12(a)], whereas the total volume is exponentially decreased [see Figure 2.12(b)], i.e., the apparent shape of an object is well approximated in the early stage of construction, but fine details are retained in the higher level.

Figure 2.13 shows the volume of intersection octants and inside octants. Since internal octants are generally not found in the early reconstruction, the graph starts from the third level of reconstruction. A convex object (e.g., a ball-shaped candle) produces the highest number of inside octants (6067 [oct]) among others, but due to the concavities, a dummy model produces 397 inside octants, which are visualised in red in Figure 2.11(b)

[5]There is an algorithm that adaptively selects the size of an initial octant [31], but it is manually selected in these experiments considering the physical size of an object.

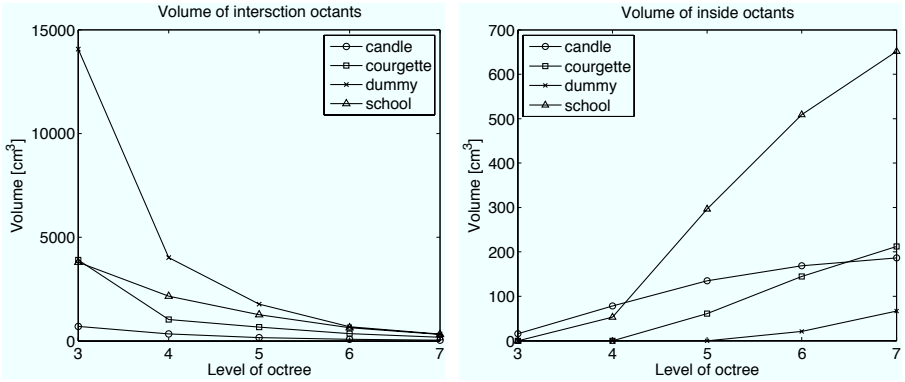


Figure 2.13: Inside and intersection volume: (a) intersection volume and (b) inside volume relative to the level of an octree.

and (f), respectively. In general, the inside volume increases proportionally to the level of an octree, while the volume of intersection octants is exponentially reduced. Thus, it is possible to use the ratio of inside and intersection octants indicate how well a VH approximates an object. For example, it is better to keep iterating the octree construction algorithm until the volume of inside octants saturates to a certain level if processing time is not a concern.

The processing time of the two methods are shown in Figure 2.14 where log scaled vertical axes have been used to illustrate the differences more significantly. Employing method II, rapid reconstructions are generally obtained [see Figure 2.14(b)] except for the dummy, which has considerably different appearance in each view so that extensive octree updating can be inevitable whenever a new image is added. Furthermore, the four limbs of the dummy create many non-convex regions in silhouettes that make the intersection test complicated. The seven-level of an octree construction times for the four objects shown in Figure 2.10(a)-(d), are 93.15, 25.1, 25.3 and 17.06[sec] when method I is applied, whereas method II requires 51.72, 23.61, 15.38 and 44.09[sec]. Thus, the method II reduces about 30% of the processing time when a simple shape of an object is used.

The progressive reconstruction results of the dummy model are shown in Figure 2.15(a)-(g), where each octant is described by 12 triangular meshes, i.e., two for each face. Since the object is largely distributed in the initial octant shown in Figure 2.15(a), the initial and second constructions have the same volume size but the number of octants

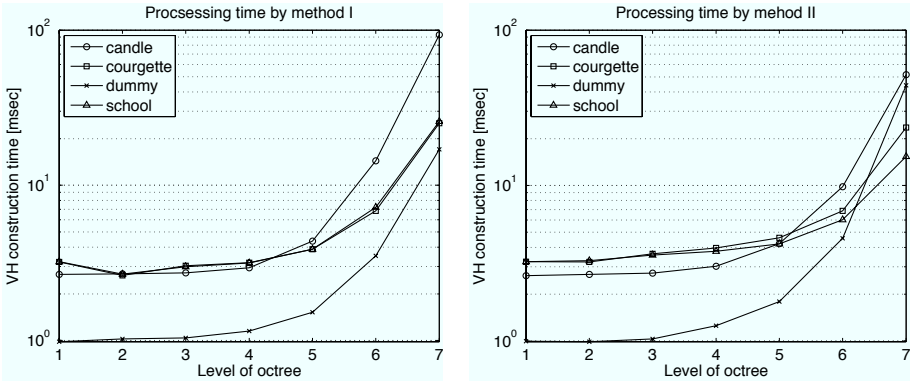


Figure 2.14: Processing time of two methods: (a) result of method I and (b) method II. The vertical axes of both graphs are log scaled.

increases by eight times in Figure 2.15(b). After a six-level of construction, the object is recognisable [see Figure 2.15(f)] and further iteration enhances more details in two wrists and ankles, as shown in Figure 2.15(g).

In SfS reconstruction, the viewing direction also affects the visual quality of a VH as well as the shape of the occluding contour. Ahuja et al. demonstrate that more than 90 per cent of an object volume is successfully estimated from the subset images of 13 orthographic viewing directions, such as the three face views, six edge views, and four corner views of upright cube [32]. Laurentini associates a convex VH from external views with a VH from internal viewing directions bounded by external VH to maximise volume confinement [14]. Furthermore, Shanmukh et al. propose a heuristic algorithm that selects the optimal set of viewing directions for the best VH [33]. Figure 2.16 illustrates the influence of viewing direction in a VH construction. Fifteen images of a school model captured at rotation angle 0° to 90° approximate the object as shown in Figure 2.16(a) where the total number of octants is 6910 and the volume is 1732.61 [cm^3] (refer Table 2.2 for more details). However, four images taken from main diagonal positions of the xy plane of $\mathcal{F}_w$ estimate a much smaller VH than the result [see 2.16(b)], and it is more similar to the result using 60 images from a full rotation [see 2.16(c)]. Therefore, silhouettes from views which surround an object generate the best volume approximation but the viewing directions should be chosen to avoid redundancy.

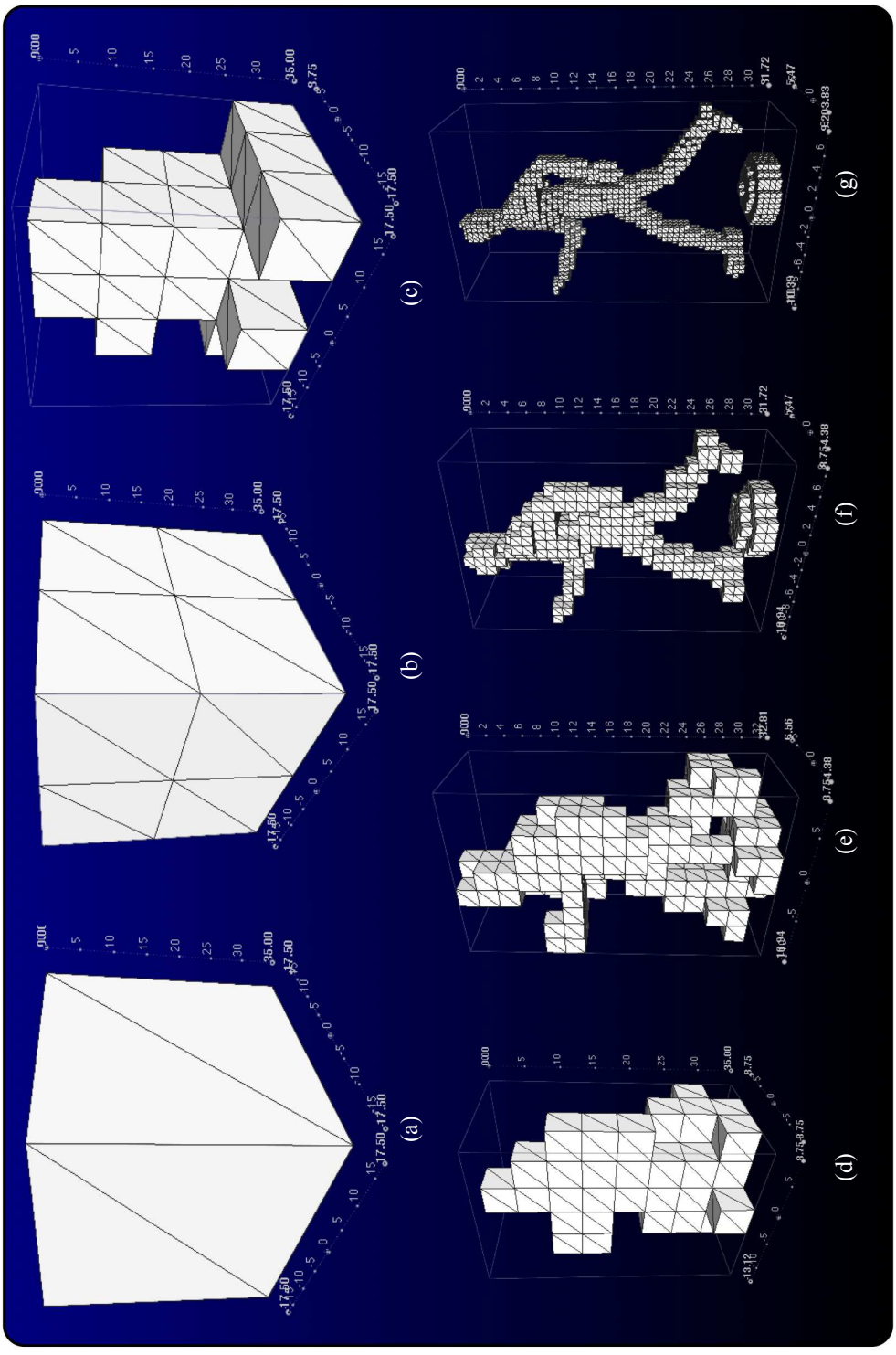


Figure 2.15: Intermediate results of the progressive reconstruction of the dummy shown in Figure 2.10(d): Each image from (a) to (g) corresponds to the 1 to 7-level of octree construction.

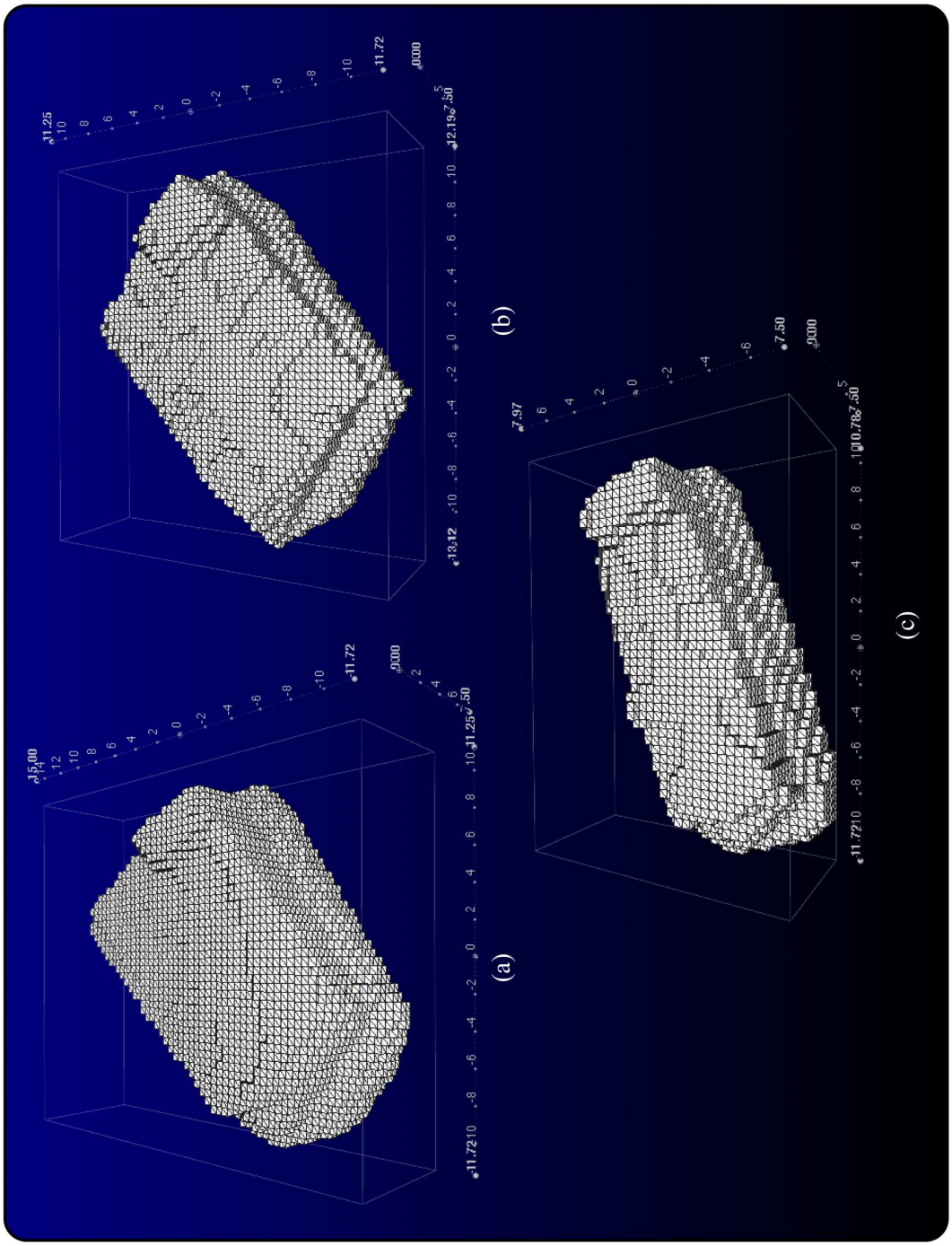


Figure 2.16: VH results relative to viewing directions: (a) result using 15 images equally spaced at a quarter of a rotation; (b) result using four images from the main diagonal directions in the xy plane of a world frame; (c) result using 60 images of the rotated object.

Table 2.2: VH relative to viewing directions.

	Intersection		Inside		Total		
	Oct.	Vol.	Oct.	Vol.	Oct.	Vol.	Proc. Time
0∼45° (7)[a]	5834	600.883	3090	1830.87	8924	2431.76	3.735
0∼90° (15)	4567	470.387	2343	1262.23	6910	1732.61	6.521
0∼180° (45)	3148	324.234	1583	710.987	4731	1035.22	10.167
0∼360° (60)	3087	317.951	1679	651.661	4766	969.612	19.17
0°, 15°, 30°, 270°	4425	455.761	2525	1070.45	6950	1526.21	1.942

[a]The number in parenthesis represents the number of images captured during rotation.
Units of Oct., Vol., and Proc. Time are [oct], [cm^3], and [sec].

2.4 SfS without an octree

2.4.1 Parallelogram and pillar representation

Although the octree structure successfully represents a VH in general cases, there are alternative methods utilising more application-specific representations. For example, assuming orthogonal projection, the intersections of back-projected rays on the same plane form parallelograms, so that the volume segments can be defined by the stack of parallelograms [17]. The algorithm incrementally enhances the volume approximation, i.e., the object volume is initially constructed from two views, followed by further refining process if more images are available. To facilitate the refining process, planes including parallelograms are sorted according to their z axis values, and each parallelogram on the xy planes is then divided by a set of lines, which are further sorted according to the x values. Therefore, when a line intersects the silhouettes, it is broken into segments, which modifies the previous y values of the line or inserts new values.

On the other hand, Niem et al. investigate a rectangular volume unit, called a pillar, which does not rely on the time-consuming intersection test that a cubical octant uses [34]. Instead of eight corners, a pillar is represented by its two ends and centre position. Thus, the intersection test only verifies whether the line, associating two projections of ends of a pillar, intersects a silhouette. A pillar corresponding to the line is broken simultaneously when the line is segmented, and a mesh growing algorithm [34] adaptively constructs surfaces from pillars, i.e., highly curved regions are described by more surface triangles than regions with low curvature.

2.4.2 3D line segment representation

Voxels can be replaced with 3D line segments, which are more memory efficient and faster than octree representation. Also, it can be used to compress the size of the final 3D data. Grau et al. exploit this representation in a broadcasting system[6] that requires to merge real and virtual scenes [35]. To create a virtual content, 3D surface models and their material properties (e.g., texture) should be prepared in advance and these virtual contents are added to the real scene contents at a normal video frame rate. However, the traditional 3D scene reconstruction method cannot satisfy this criterion. For fast 3D reconstruction, Euclidean three-space is samplled by a set of line segments, which are projected on to the images. Finally, a modified marching cubes algorithm generates surfaces from the line segments.

Another research on 3D line segment introduced by Fang et al. [36], is similar to the traditional line segment-based volume representation except that it includes the dense reconstruction algorithm and a unique meshing method called a data conversion algorithm. For example, the method initially approximates the shape of an object using 3D line-based model followed by the dynamic line resolution adjustment, which inserts additional line segments according to the shape of a triangle in order to enhance the quality of the visualisation. However, these methods are basically equivalent to the octree method with regard to projecting approximately known 3D object positions onto the 2D images.

2.5 Visual hull from colour information

The general SfS methods extract 3D information from a set of 2D binary images called silhouettes. Although the use of binary images makes a reconstruction process more robust against image noise and colour ambiguity, it is not an appropriate choice in terms of photo-realistic shape reconstruction, i.e., it loses colour information. Shape from photo consistency assumes a complete scene model includes not only surface geometry but also surface reflectance models and scene illumination [37], and this type of reconstruction

[6] Although passive methods are not currently in common use in a studio environment, due to their accuracy, robustness and difficulty of real time operation, they realise 3D rendering from multiple cameras.

intends to preserve the reproduction consistency.

In fact, a conventional stereo camera system has already used photometric features, e.g., corners or edges. However, feature correspondence is required before triangulating two photometric features. In 1996, Collins proposed a method which exploits photometric information but does not require feature correspondence in the plane-sweeping algorithm [38]. Afterward, Seitz et al. proposed two conditions for the photorealistic 3D reconstruction and the special camera configuration for their photorealistic reconstruction called Voxel Coloring (VC) [39]. Kutulakos et al. extend VC to address arbitrary camera position, which is called Space Carving (SC) [37]. Kutulakos also analysed influence of noise on SC algorithm and proposed the robust SC algorithm [40].

2.5.1 Plane sweeping algorithm

The plane sweeping algorithm considers the volume reconstruction from aerial images as multiple stereo reconstructions. The algorithm needs to determine corresponding features across views, which have no evidence to track features because of wide distribution of views unlike motion sequence [38]. Collins [38] suggested three constraints: matching algorithm should be applied to any number of image greater than two; the complexity of algorithm is linear, i.e, $O(n)$, where n is the number of images; all the images have to be treated equally, i.e, there is no special pre-process or weighting for the reference image. Most stereo matching techniques search corresponding features using the epipolar constraint so that the complexity of matching algorithm for the all image pairs is $O(n^2)$. Other multiple image matching algorithms need templates from a reference image. In this aspect, the plane-sweeping algorithm is superior to traditional stereo matching algorithm.

The algorithm initialises the 3D plane partitioned into cells and it then sweeps the plane in the perpendicular direction to the initial plane. When sweeping, it premises that the area which has the most viewing rays are highly like to be a correct 3D. The size of cell defines the finest resolution for reconstruction. Since the number of viewing ray intersecting the cell is counted at each sweeping procedure, the algorithm demands large memory capacity to keep track of the intersection if a fine cell is used. For the efficient sweeping and back-projection of features, each move of sweeping is implemented by a planar homography.

2.5.2 Voxel colouring

Photorealistic 3D reconstruction can be achieved when two criteria are met such as photo integrity and broad viewpoint convergence [39]. Since the former criterion states that the input image should be reproduced by re-projecting the estimated 3D results onto the viewpoint, the accurate and dense 3D surface vertices associated with texture colour are required. On the other hand, the second criterion requires an accurate re-projection over a wide range of target viewing points, so that integrating widely-distributed images is required. Although previous feature and contour-based reconstruction methods can estimate texture map, the resulting accuracy relies on the accuracy of the feature detection and matching. Furthermore, largely distributed images result in significant matching ambiguity in existing matching methods.

Therefore, Seitz et al. utilised invariant colour instead of invariant shape to solve the correspondence problem. Colour invariant point does not require that the point be contained in every consistent scene. Consequently, all of these colour invariant points are used in the matching process. However, assigning a unique colour to one voxel located in the n-by-n-by-n 3D volume from 2D images is an ill-posed problem. To make it tractable, an additional constraint called ordinal visibility constraint, is employed and it defines a non-negative function, which can distinguish two occluded points by means of the distance from a viewing plane defined by every viewing point. Therefore, special camera configuration is required, e.g., overhead inward-facing camera or outward-facing camera distributed around a sphere. The VC algorithm projects every voxel in each layer of the 3D volume onto all image planes. The similar-colour pixels in all viewing points are then collected by thresholding their colour correlation values, i.e., if the pixel colour is similar then these points correspond to the voxel. The same algorithm is then applied to the next layer.

The VC can cope with occlusions according to the ordinal visibility constraint, and it includes modelling of surface texture. Due to the explicit modelling of the occlusions, cameras can be located far apart than each other, and multiple images are used for dense reconstruction without degrading the accuracy of the scene reconstruction. However, special assumptions are required, for example, Lambertian opaque object, uniformly

distributed lighting condition, and a special camera configuration.

2.5.3 Space carving

Although VC works well under controlled camera configuration, it is not applicable when cameras are arbitrary distributed. SC is basically an extension of VC. For example, it carves the initial volume iteratively until the initial scene converges to the Photo Hull (PH) [37], and the decision to carve is made based on the photo consistency, i.e., it needs to estimate the standard deviation of colours at the projections of voxels. Thus, VC assumes that the colour of light reflected from a single point along different directions is not arbitrary, so that it is a locally computable model.

The main difference of SC to VC is that SC can keep track of scene visibility for all input cameras in every iteration. Although this makes the updating procedure more complicated than VC, where a special camera configuration solves this problem, this capability allows multiple plane-sweeping algorithms for arbitrary camera configuration. For instance, a solid block of voxels is iteratively carved away by sweeping six planes whose sweep directions correspond to the six principle directions, and the visibility order is estimated from six sweep planes. However, SC also assumes Lambertian model, which means its performance is sensitive to the illumination condition, and a rule to define the initial volume is not suggested. SC consumes much computational time, e.g., results in [37] show 250 minutes are taken when carving initial 51 million voxels to 215 thousand. Furthermore, a sufficiently large number of photographs are required to ensure the algorithm output closely resembles the shape of a complicated scene.

2.5.4 Pros and cons

Although VC and SC achieve photo-realism (e.g., the texture information is also estimated during volume estimation), there are many possibilities that the resulting PH contains error. Dyer summarised these problems as follows [41]. First, the accuracy of PH depends on the degree of surface reflectance function. Second, the use of discrete voxel can cause error when estimating surface orientation and illumination. Third, the discrete voxel resulted in aliasing error. Finally, inaccurate threshold level to determine

photo-consistency leads to voxel classification error. Therefore, most of the latest papers of SC try to tackle these points. For example, Kutulakos proposed approximate space carving, which is correct for arbitrary discrete scenes [40]. This algorithm can cope with an unknown and arbitrary Lambertian scene that are defined by a finite set of voxels.

2.6 Conclusions

This chapter introduces the general concept of SfS for volume reconstruction. SfS does not require point correspondences used in traditional stereoscopic techniques, but it requires multiple images that are generated from views surrounding an object with known projection transforms. As a representation of a VH, an octree of voxels are normally utilised. Two octree construction methods for a VH are proposed with pseudo codes. The experimental results show that method II reduces the processing time and peak memory usage when an object has simple shape but method I performs better in case the excessive octree updating is required.

Chapter 3

Projection transform estimation from circular motion

3.1 Introduction

The accuracy of a volume reconstruction is dependent on that of the projection matrices associated with different views, which are estimated in the camera calibration process. Provided with n views in a SfS system, the calibration process should estimate $11 \times n$ parameters. However, it is a cumbersome process to locate accurate 3D-to-2D point correspondences in each view for the calibration. Instead, assuming that the intrinsic parameters of the projection matrices are identical, the dimension of the parameter space is reduced to $5 + (6 \times n)$, i.e., 5 DoF of camera characteristics and 6 DoF from a camera motion per view. Therefore, the SfS systems endeavour to analyse a 3D camera motion from n views which facilitates the calibration process.

In terms of a screw decomposition, any 3D camera motion is modelled by a pure rotation around a fixed screw axis together with translation of the screw axis [42]. For example, if an object is placed on a turntable and an image is captured by a camera at fixed position for every constant rotation of the turntable, it is possible to model its 3D

motion as a pure rotation called a circular motion[1]. A circular motion embeds many useful geometric constraints which provide significant clues for the estimation of extrinsic parameters, e.g., the fundamental matrix of every two views [43], and parameterised projection matrices in terms of a rotation angle [42]. However, any physical imperfection of the rotation and measurement error often invalidate the use of these constraints.

To obtain a more precise projection matrix in SfS, one approach refines the initial projections by minimising the errors between true silhouettes and the re-projection of an initial 3D approximation generated from inaccurate projection matrices by a SfS technique [44]. Another method exploits a tangent error, which is an absolute difference of the angles defined between a projected silhouette cone and a tangent cone, to minimise the computational expenses of an initial reconstruction [45]. However, it still requires to project all rays of a silhouette cone onto an adjacent view to define the tangent error, even though an initial reconstruction is not necessary. Recently, Wong et al. investigate frontier points in adjacent images and propose an algorithm that modifies rotation angles of views in a circular motion by minimising the re-projection error of two outer epipolar tangents [19]. However, since the algorithm assumes a perfect circular motion, the input images should be rectified if they do not satisfy with constraints of a circular motion, e.g., an image of a screw axis and an image of horizontal line remain unchanged throughout the rotation.

In practice, a pure circular motion is not easy to be realised, i.e., most circular motions behave like an approximate circular motion, where most of the motion can still be modelled as a pure circular motion but some views have a small amount of additional 3D motion. Since the motion associated with turntable image sequences in a SfS is generally more closer to an approximate circular motion than a pure circular motion, this chapter proposes an algorithm which modifies the estimated projection matrices to increase the accuracy of the object reconstruction.

The remainder of this chapter is organised as follows. Section 3.2 reviews a fundamental idea of modelling a projection matrix, and its estimation and evaluation technique. Section 3.3 presents more details of circular motion and an approximate circular motion,

[1]Alternatively, a circular motion is also achieved when a camera rotates around an object at a fixed position.

in addition to useful constraints derived from fixed entities of a circular motion. Section 3.4 presents a method which modifies projection matrices, estimated in a circular motion, for an approximate circular rotation. Experimental results of modified projection matrices and the resulting volume reconstructions are presented in Section 3.5, and finally Section 3.6 concludes this chapter. Thus, this chapter primarily investigates following aspects:

- conventional linear calibration model and its variation;
- non-linear optimisation method to obtain accurate calibration matrix;
- multiple camera calibration exploiting a camera motion;
- refined multiple camera calibration from an approximated circular motion.

3.2 Camera calibration

3.2.1 Projection models

Suppose that a camera frame $\mathcal{F}_c = (\vec{o}_c, \vec{i}_c, \vec{j}_c, \vec{k}_c)$ coincides with a world frame $\mathcal{F}_w$, an image plane $\vec{\pi}^c$ is placed in the positive side of $\vec{k}$ axis, and the normal vector of the plane $\vec{\pi}^c$ is parallel to $\vec{k}$. Then, according to the pinhole camera geometry, an image of a point placed in front of $\vec{\pi}^c$ is inverted and magnified by f/z, where a focal length f is measured by a orthogonal distance from $\vec{o}$ to $\vec{\pi}^c$, and z is the depth value of a point measured in the $\vec{k}$ direction. Therefore, a 3D-to-2D point correspondence $\vec{x}^c \mapsto \vec{x}^m$ is related by

$$\vec{x}^m = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \vec{x}^c, \tag{3.1}$$

where $\mathcal{F}_m$ denotes a 2D image frame. In other words, an image of an object is magnified as an object moves toward a camera. Furthermore, two points lying on the same line in a 3D space have different scaling values depending on their depth position (e.g., the projection of $\vec{x}^c = [x\ y\ z\ 1]^{\mathrm{T}}$ produces $\vec{x}^m = [fx/z\ fy/z\ 1]^{\mathrm{T}}$), which lead to the projective distortion in a 2D image plane (e.g., images of two parallel lines converge on a point called a vanishing

point). In some cases, however, fixed magnification regardless of the depth value is more preferred. For example, a weak perspective projection presumes an object is placed at infinite distance from a camera, so that all 3D points have fixed scaling ($f/z > 1$), and an orthographic projection further simplifies a projection model by assuming that an image of an object is not inverted in $\vec{\pi}^c$ with a unit scaling ($f/z = 1$). Although both projections do not suffer from projective distortion, they are affected by up to an affine distortion. Therefore, a weak perspective projection and an orthographic projection are also referred to as affine projections [3].

For a digital camera, an image plane $\vec{\pi}^c$ is replaced with a Charge-Coupled Device (CCD) sensor, and a lens is installed to concentrate rays, which produce brighter image[2]. However, simple lenses suffer from a number of aberrations [1], and a CCD can distort an image in accordance with its mechanical fault or physical sensor characteristics [3]. For example, radial distortion (e.g., pincushion distortion and barrel distortion) is often observed in an image due to lens aberrations, which requires additional non-linear terms[3] in the linear projection model of (3.1) [1, 9]. Also, a CCD sensor may have colour and geometrical distortion, e.g., the centre of an image is moved by $[c_x\ c_y]^{\mathrm{T}}$, where the scaling values are not identical in the both axial direction, and a CCD can be skewed by s_k. To account for these effects, (3.1) can be revised as

$$\vec{x}^m = \begin{bmatrix} f_x & s_k & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \vec{x}^c, \tag{3.2}$$

where f_x and f_y denote the focal length of a camera in terms of pixel dimension in the x and y direction [3]. Thus, if γ_x[px]/[cm] represents a unit ratio which converts real world measurements to a pixel unit in the x direction, then the focal length in the x direction is $f_x = f \times \gamma_x$. Similarly, $f_y = f \times \gamma_y$, $c_x = u_x \times \gamma_x$, and $c_y = u_y \times \gamma_y$, where $[u_x\ u_y\ 1]^{\mathrm{T}}$ is the optical centre of $\vec{\pi}^c$.

Since the projection matrix (3.2) assumes that a camera frame coincides with a world frame, it should be re-written if $\mathcal{F}_c$ is rotated and translated from a world frame,

[2]The amount of rays are controlled by an aperture, where the exposure time can be adjustable.
[3]A non-linear camera model is beyond the scope of this research, but some details are found in [46, 47].

i.e.,

$$\vec{x}^m = \begin{bmatrix} f_x & s_k & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{\text{int}} & -R_{\text{int}}\vec{o}_c \end{bmatrix} \vec{x}^w, \tag{3.3}$$

where the matrix R_{int} is a 3-by-3 rotation matrix between $\mathcal{F}_w$ and $\mathcal{F}_c$, and $\vec{o}_c$ represents a camera centre with respect to the world frame. Since the first matrix in (3.3) is comprised of camera parameters whilst the second matrix is only related to 3D camera motions, the first is often called an intrinsic calibration matrix and the second is called an extrinsic calibration matrix.

3.2.2 Camera calibration

At least six 3D-to-2D point correspondences are required to estimate the linear projection matrix given in (3.3), because it has 12 unknown variables[4] and each corresponding point pair produces two equations. However, a rule of thumb is that for a good estimation the observed data should exceed the unknown variables by a factor of five, i.e., good approximation is expected when around 30 point pairs are available [3]. Although there are various methods for the accurate estimation of a projection matrix [9, 46, 47, 48], the fundamental two-step calibration, consisting of a linear and non-linear optimisation, is generally utilised for the volume reconstruction in this thesis.

As a linear solution of (3.3), a projection matrix should minimise the re-projection error,

$$r_i(P) = \vec{x}_i^m - P\vec{x}_i^w, \tag{3.4}$$

where a subscription i represents the index of a corresponding pair, i.e., $\vec{x}_i^w \mapsto \vec{x}_i^m$, and P is a projection matrix which is a product of an intrinsic matrix and an extrinsic matrix. Thus, a cost function of an optimised projection matrix P is

$$P = \arg\min_P \{\sum_i |r_i(P)|^2\}. \tag{3.5}$$

[4]With an assumption of a unit scaling, the number of unknowns can be reduced to 11 parameters.

Equation (3.5) is algebraically rearranged as a linear system of equations to give

$$A\vec{p} = \begin{bmatrix} \vec{0}^{\mathrm{T}} & -s_i(\vec{x}_i^w)^{\mathrm{T}} & v_i(\vec{x}_i^w)^{\mathrm{T}} \\ -s_i(\vec{x}_i^w)^{\mathrm{T}} & \vec{0}^{\mathrm{T}} & u_i(\vec{x}_i^w)^{\mathrm{T}} \\ -v_i(\vec{x}_i^w)^{\mathrm{T}} & u_i(\vec{x}_i^w)^{\mathrm{T}} & \vec{0}^{\mathrm{T}} \end{bmatrix} \vec{p} = 0, \tag{3.6}$$

where $\vec{x}_i^m = [u_i \ v_i \ s_i]^{\mathrm{T}}$ and P is vectorised as $\vec{p}$. Since (3.6) is a homogeneous system, a solution exists in a null space of A, which is normally obtained by the eigen analysis of a matrix A. This linear solution for a projection matrix is first introduced in [49], where it is called a solution of Direct Linear Transformation (DLT). Since a DLT solution is the linear model regression from observations (i.e., 3D-to-2D point correspondences), a result is sensitive to the accuracy of a 2D point detection algorithm and data distribution, e.g., a DLT solution degenerates if all 3D points exist on the same plane. In order to provide a sufficient number of accurate feature points from different planes in a 3D space, a 3D calibration rig attached with a checkerboard pattern is normally exploited in an offline calibration process. A square of the checkerboard pattern facilitates the 2D feature detection process. Moreover, it also enhances the accuracy of the data. Some examples of calibration patterns are shown in Figure 3.1(a) and (b).

An initial linear solution can be further improved by the Levenverg-Marquardt (LM) non-linear least square optimisation [50], which iteratively searches the best solution that fits given observations with an initial guess obtained by the DLT algorithm. Using q pairs of observations, each re-projection error given in (3.4) can be represented as a residual error vector $\vec{r}$, i.e.,

$$\vec{r}(\vec{p}_k) = [r_1(\vec{p}_k) \ \cdots \ r_q(\vec{p}_k)]^{\mathrm{T}}, \tag{3.7}$$

where $\vec{p}_k$ is a vectorised projection matrix in the k-th iteration. Thus, the best estimation $\vec{p}_{\mathrm{best}}$ should minimise the norm of the residual error vector, $|\vec{r}(\vec{p}_{\mathrm{best}})| \leq \epsilon_r$. To search for $\vec{p}_{\mathrm{best}}$, the LM algorithm iteratively updates a current solution (i.e., $\vec{p}_{(k+1)} = \vec{p}_k + \delta\vec{p}_k$) in a 12-dimensional parameter space until it reaches a local minimum of the cost function given in (3.5). Thus, an efficient and stable determination of $\delta\vec{p}_k$ is a major concern of the LM algorithm, and it is implemented by computing a gradient vector of the cost function

with a regularised parameter.

The Taylor series of a residual error function of (3.4) at an updated parameter vector $\vec{p}_{(k+1)}$ is

$$r_i(\vec{p}_{(k+1)}) = r_i(\vec{p}_k) + \left\{ \delta p_1 \frac{\partial r_i(\vec{p}_k)}{\partial p_1} + \cdots + \delta p_{12} \frac{\partial r_i(\vec{p}_k)}{\partial p_{12}} \right\} + O(n^2), \tag{3.8}$$

where $O(n^2)$ denotes the higher order terms. Thus, the residual error in the next iteration can be approximated by the current error without higher order terms of the Taylor series. Suppose that a gradient vector with respect to the parameter vector $\vec{p}$ is defined as $\nabla_{\vec{p}} = [\frac{\partial}{\partial p_1}, \cdots, \frac{\partial}{\partial p_{12}}]^{\mathrm{T}}$. Then (3.8) is re-written as

$$\vec{r}(\vec{p}_{k+1}) \simeq \vec{r}(\vec{p}_k) + (\nabla_{\vec{p}} \vec{r}(\vec{p}_k)^{\mathrm{T}})^{\mathrm{T}} (\delta \vec{p}_k), \tag{3.9}$$

where $(\nabla_{\vec{p}} \vec{r}(\vec{p}_k)^{\mathrm{T}})^{\mathrm{T}}$ is called a Jacobian matrix and denoted as J. Therefore, $\delta \vec{p}_k$ should satisfy $\vec{r}(\vec{p}_k) + J\delta \vec{p}_k = 0$. If a Jacobian matrix is not singular, an increment of the iteration is easily determined as $\delta \vec{p}_k = -J^{-1} \vec{r}(\vec{p}_k)$. However, when J is a rectangular matrix, a pseudo inverse of J is used for the stable estimation of $\delta \vec{p}_k$. Furthermore, the LM algorithm has a regularising parameter β_k to avoid a singular matrix and to provide faster convergence in case of an over-parameterised problem [3], i.e.,

$$\delta \vec{p}_k = -(J^{\mathrm{T}} J - \beta_k I)^{-1} r(\vec{p}_k), \tag{3.10}$$

where $J^{\mathrm{T}} J$ is called the Hessian matrix and I is an identity matrix. If the computation of a Jacobian matrix is straightforward, an explicit J is directly provided to the LM iteration. However, the numerical approximations of partial differences are generally replaced with the explicit J (see forward differences formulae in [51]). In this thesis, a built-in MATLAB function, $lsqnonlin(\cdot)$, is used for a LM optimisation.

Some examples of performance evaluations are illustrated in Figure 3.1. Three images of two type of 3D calibration rig are captured with different setting of internal and external camera parameters [see Figure 3.1(a)-(c)]. The first image has a 4×4 checkerboard pattern in a 19[cm]×19[cm] face, whilst more refined pattern (i.e., a 6×6 in a 18[cm]×18[cm] face) is attached to each face of a calibration rig in other two images.

Thus, 43 3D-to-2D point correspondences are detected by the Harris corner detector [52] in Figure 3.1(a), and 91 pairs are detected in other two (see +'s in the images).

A 3D calibration rig represents a 3D world frame, i.e., the left face of the rig corresponds to the yz plane of $\mathcal{F}_w$ and the right face is the xz plane. Apart from the first two images, the correspondence pairs in Figure 3.1(c) are not equally distributed, i.e., the 2D points in the yz plane are more condensed than the other face. To compare the performance of the estimated projection matrices from different observations, an average re-projection error is evaluated, i.e., the performance is better, if

$$error(P) = \frac{1}{q}\sum_{i=1}^{q} |\vec{x}_i^m - P\vec{x}_i^w|^2 \tag{3.11}$$

is smaller. Figure 3.1(d) shows the average re-projection errors of projection matrices with respect to the number of correspondence pairs. Since a DLT solution of a projection matrix can be achieved if there are at least six point pairs, the performance test is undertaken by varying the number of point pairs from 6 to 43, which are selected randomly to show the calibration performance relative to the quantity of observations. A linear DLT solution of six point pairs almost perfectly fits to the observations (see the small error in the beginning of the error graph). However, the re-projection error rapidly increases until it saturates after around 25 point pairs, because a DLT solution is the only best solution in the least square sense. The final re-projection errors of DLT solutions of (a), (b), and (c) are all smaller than 2[px], i.e., 1.6925[px], 1.4863[px], and 1.2499[px], respectively. This error range is acceptable when considering the size of images (e.g., 1024×1024, 640×480, and 1024×1024), and these errors are caused by the 2D feature detection and an imperfect calibration pattern. The LM optimisation further refines these solutions to 1.6935[px],1.4850[px] and 1.2454[px]. The internal and external parameters extracted from each of the calibration matrices from DLT and LM method are listed in Table 3.1 and 3.2, respectively.

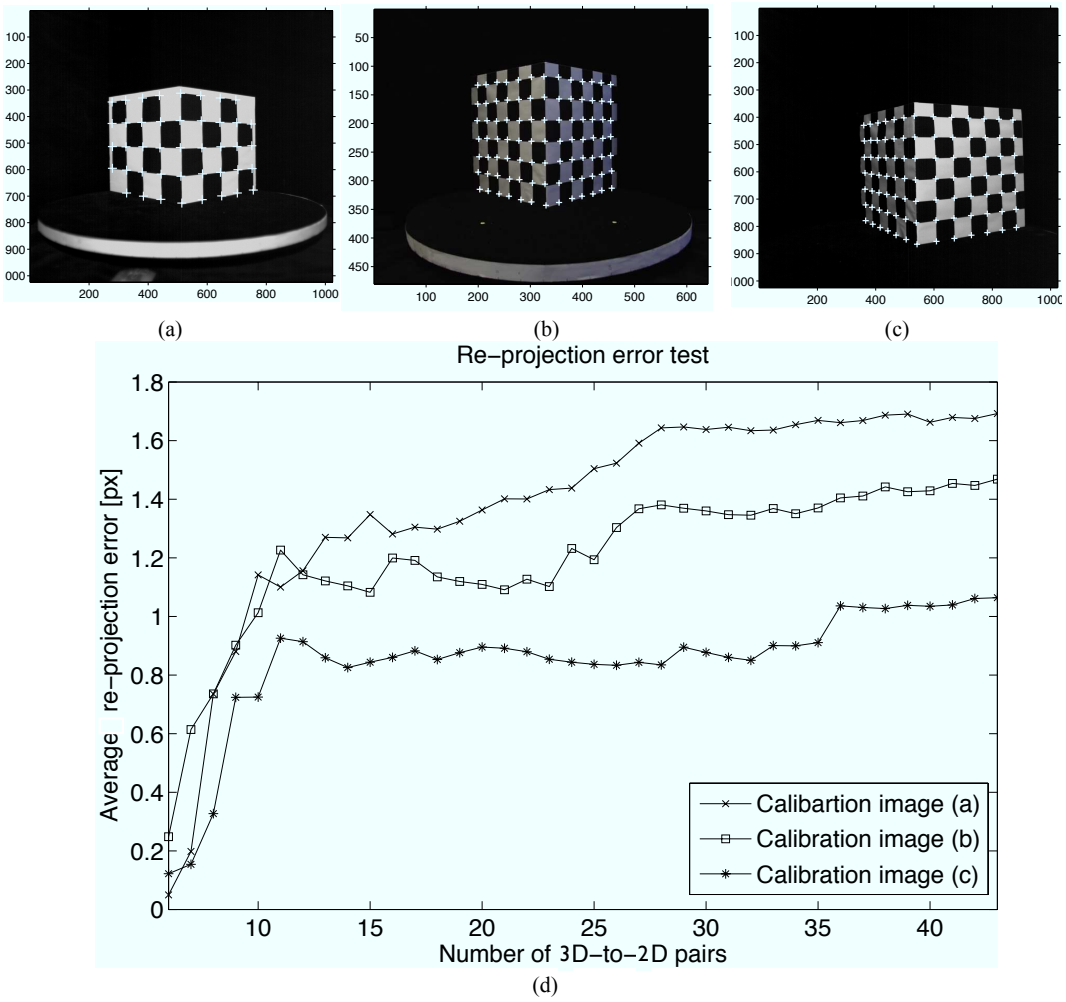


Figure 3.1: Calibration performance: (a)-(c) three test images of a calibration pattern, where + denotes a 2D point used for the calibration; (d) average re-projection errors of DLT solutions obtained from (a), (b) and (c).

Table 3.1: Camera parameters from DLT solution.

	Internal parameters					External parameters			
	f_x	f_y	s_k	c_x	c_y	t_x	t_y	t_z	$error(P)$
(a)	1172.08	1164.90	12.79	533.69	460.33	-36.79	-38.26	11.30	1.693
(b)	603.76	605.23	4.21	313.46	241.25	-34.08	-33.10	11.48	1.486
(c)	1159.31	1156.58	7.86	544.91	509.14	-21.91	-38.21	13.10	1.250

Table 3.2: Camera parameters from LM solution.

	Internal parameters					External parameters			
	f_x	f_y	s_k	c_x	c_y	t_x	t_y	t_z	error(P)
(a)	1172.41	1164.68	12.59	533.33	460.81	-36.79	-38.26	11.30	1.694
(b)	601.52	602.09	3.92	316.43	239.55	-33.84	-32.99	11.47	1.485
(c)	1158.84	1155.25	7.62	545.64	509.14	-21.88	-38.19	13.07	1.245

3.3 Circular Motion

3.3.1 Projection matrix and cost function of SfS

When a view is related to a reference view by an unknown 3D rotation matrix R without changing the internal camera parameters, its projection matrix can be derived from a projection matrix at the reference position. For example, if the projection matrix at a reference position is given as a general projection matrix in (3.3), then the projection matrix of a rotated view is given by

$$\begin{aligned} P(\vec{\theta}) &= KR_{\text{int}}[R(\vec{\theta}) \ -\vec{t}] \\ &= KR_{\text{int}}[R_x(\theta_x)R_y(\theta_y)R_x(\theta_z) \ -\vec{o}_c], \end{aligned} \tag{3.12}$$

where K denotes an intrinsic calibration matrix and $\vec{\theta}$ represents a vector associated with rotation angles in each basis of $\mathcal{F}_w$, i.e., $\vec{\theta} = [\theta_x \ \theta_y \ \theta_z]^{\mathrm{T}}$. Thus, a rotation matrix $R(\vec{\theta})$ is

$$R(\vec{\theta}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} \cos(\theta_x) & \sin(\theta_x) & 0 \\ -\sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.13}$$

This idea can be utilised in parameterising projection matrices for a pure circular motion.

One of the strong constraints imposed by a circular motion is that the internal parameters remain identical over the whole image sequence. Therefore, K, R_{int} and $\vec{o}_c$ in (3.12) are identical for all projection matrices in a pure circular motion. Also, if it is assumed that a screw axis coincides with the z axis of a world frame, the R_x and R_y terms are removed from $R(\vec{\theta})$, i.e., a projection matrix is parameterised by a rotation angle of the z axis, i.e.,

$$P(\theta_z) = K R_{\mathrm{int}} \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 & \\ -\sin\theta_z & \cos\theta_z & 0 & -\vec{o}_c \\ 0 & 0 & 1 & \end{bmatrix}, \tag{3.14}$$

where θ_z is a rotational angle measured from a reference position in the clockwise direction, and $\vec{o}_c$ represents the position of the camera centre when $\theta_z = 0$. Therefore, if an offline camera calibration is performed at a reference position, K, R_{int} and $\vec{o}_c$ in all projection matrices are determined in a turntable image sequence. Moreover, if the rotation angles are given as a constant over all images, any further calibration process is not required.

When a projection matrix is parameterised as above, a cost function (2.7) for the volume reconstruction is also revised in terms of a rotation angle, i.e., a status of a 3D vertex point $\vec{x}^w$ in the i-th image is classified by

$$cost_{\mathrm{p}}(\vec{x}^w, \theta_i) = \begin{cases} 1 & \text{if } P(\theta_i)\vec{x}^w \in \mathcal{S}_i \\ 0 & \text{otherwise} \end{cases}, \tag{3.15}$$

where $\mathcal{S}_i$ is an object silhouette in the i-th image and θ_i is the i-th rotational angle from the reference position. Thus, given n images the point is classified as outside when

$$\prod_{i=0}^{n} cost_{\mathrm{p}}(\vec{x}^{w}, \theta_i) = 0, \tag{3.16}$$

where n is the total number of silhouette images. In most cases, $R_x(\theta_x) = R_y(\theta_y) = I$ and $\vec{t}$ is fixed in a turntable image sequence, i.e., $P(\theta_z)$ in (3.14) is acceptable. However, when θ_z is not accurately measured (i.e., $\hat{\theta}_z = \theta_z + \epsilon_{\mathrm{a}}$) or a turntable is wobbly (i.e., $|\theta_x| + |\theta_y| > 0$), the re-projection error increases and this error propagates to the next rotation, which consequently affects the volume reconstruction and surface construction.

3.3.2 Fixed entities in a circular motion

In a circular motion, two lines and three points are invariant to the rotation angle, i.e., they have fixed positions in every image plane, as illustrated in Figure 3.2. The first fixed entity, a vanishing line ($\vec{l}_h^m$) of the xy plane ($\vec{\pi}_{xy}^m$), is defined by two vanishing points in the x and y direction. A vanishing point in the x direction is found at a point where two lines parallel to the x axis converge (see $\vec{v}_x^m$ in Figure 3.2). Similarly, $\vec{v}_y^m$ is obtained in the y direction. In a pure circular motion, the vanishing line $\vec{l}_h^m$ also represents an image of a plane defined by all camera centres. Therefore, the epipoles of adjacent images should lie on this line (see two adjacent epipoles of the i-th view, e.g., $\vec{e}_{i-1}^m$ and $\vec{e}_{i+1}^m$). Consequently, $\vec{l}_h^m$ should be fixed over all images. The other fixed line is an image of rotation axis. Since the z axis coincides with the a rotation axis by assumption, an image of the z axis ($\vec{l}_s^m$) is fixed in all views. In addition, three fixed points are located on $\vec{l}_s^m$: the intersection point ($\vec{x}_a^m$) of two invariant lines (l_s^m and l_h^m), a vanishing point in the z axis ($\vec{v}_z^m$), and an image of the world origin ($\vec{o}^m$).

These fixed entities are also related to a camera calibration matrix. If a vanishing point along the x direction is represented as $\vec{v}_x^w = [1\ 0\ 0\ 0]^{\mathrm{T}}$ in $\mathbb{P}^3$, then an image of this vanishing point is found in the first column vector of a calibration matrix P. Similarly, the second and the third column vector of P indicate vanishing points in the y and z

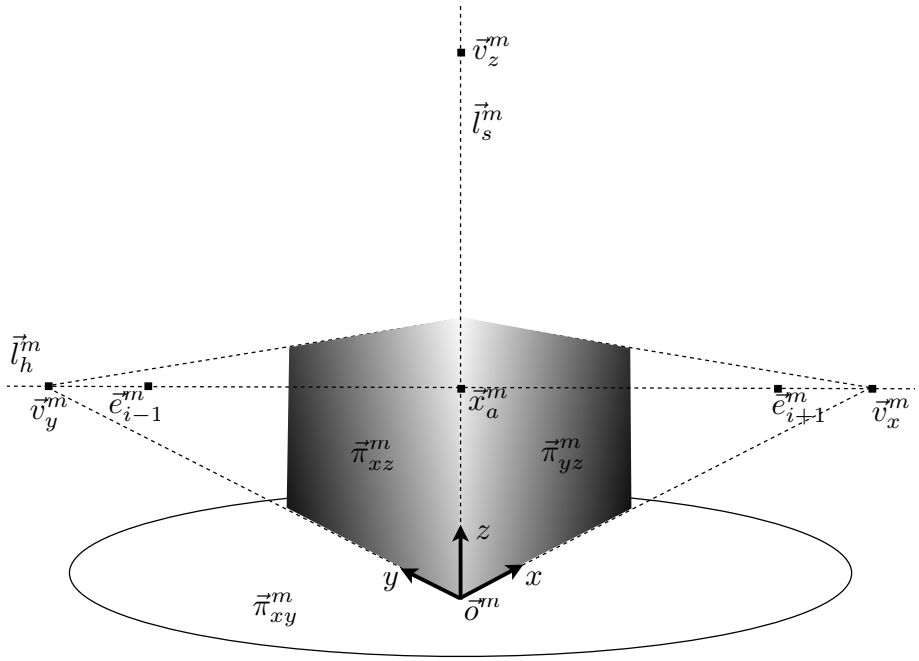


Figure 3.2: Example of fixed entities of a circular motion: a vanishing line of a xy plane ($\vec{l}_h^m$) and an image of a screw axis ($\vec{l}_s^m$) are unchanged over all images in a circular motion. Additionally, three points on a rotation axis are also invariant, e.g., $\vec{v}_z^m$, $\vec{x}_a^m$ and $\vec{o}^m$.

direction. Therefore, $\vec{l}_h^m$ is described as

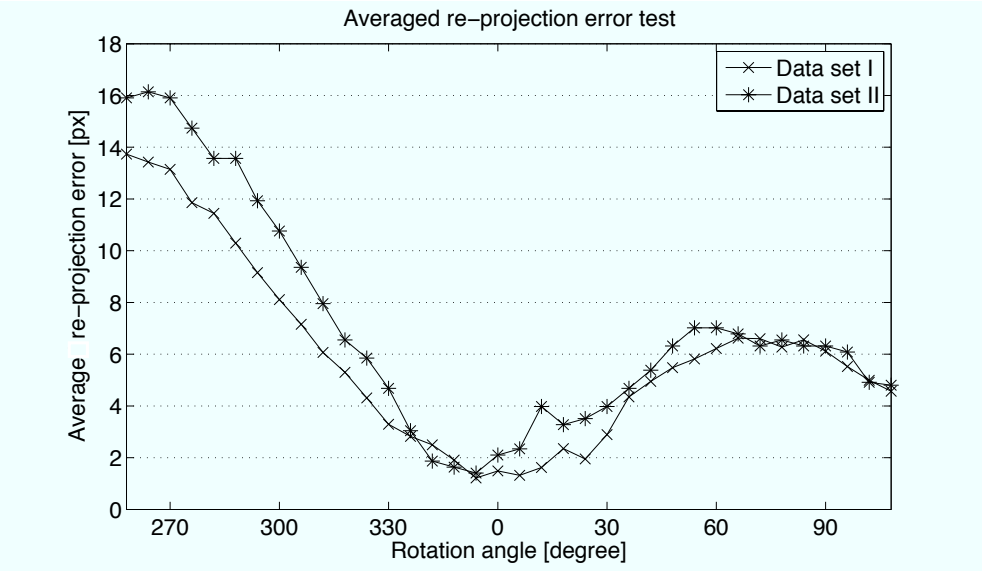
$$\vec{l}_s^m = [\vec{p}_1]_\times \vec{p}_2, \tag{3.17}$$

where $\vec{p}_i$ is the i-th column vector of the projection matrix. The image of the screw axis (i.e., a line $\vec{l}_s^m$) is defined by the cross product of an image of the world origin and a point at infinity in the z directions, i.e.,

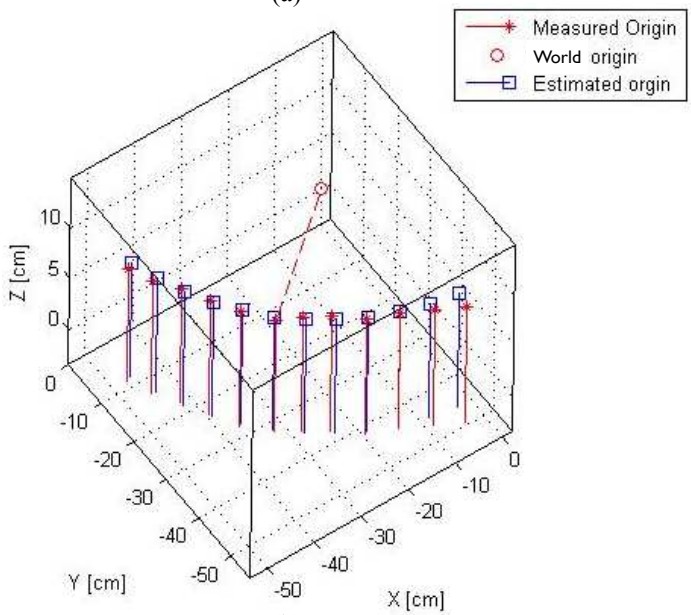
$$\vec{l}_v^m = [\vec{p}_3]_\times \vec{p}_4. \tag{3.18}$$

Although these two lines should be identical for all images in a turntable sequence under the assumption of a circular motion, the constraints are easily violated when there is some perturbation of the screw axis due to physical imperfection of the turntable.

Figure 3.3(a) shows the average re-projection errors of projection matrices estimated from a circular motion with respect to a rotation angle. Two different calibration



(a)



(b)

Figure 3.3: Example of estimation error of a circular motion: (a) average re-projection error of projection matrices estimated from a pure circular motion with respect to rotation angle; (b) true * and estimated camera centres □ in a world frame, where the reference camera position is connected to the origin of world frame ∘.

patterns (i.e., data set I and II) are used to show the estimation error in practice. The data set II is obtained from a sparse calibration pattern shown in Figure 3.1(a), whilst data set I exploits more refined pattern shown in Figure 3.1(b). The estimated projection matrices around $\pm 30^\circ$ from a reference position are within the reasonable error bound from 1.21[px] to 3.29[px] in data set I, and from 1.40[px] to 4.68[px] in data set II. However, the re-projection error increases as θ_z increases because the error propagates to images at higher rotation angles. Thus, errors in the left side of the origin is normally higher than the right if a turntable rotates in the clockwise direction. This error is not only due to the imperfect θ_z, but also due to a hidden 3D translation which is omitted in the assumption. To show the actual 3D translation, true camera centres (marked as *) and estimated centres (marked as $\square$) of data II are visualised in Figure 3.3(b), where the reference position is connected to the origin of the world frame by a dotted line. It shows the true camera centres oscillate its position from the estimations. Therefore, it seems reasonable to consider the turntable motion as an approximate circular motion. However, it is difficult to estimate these additional 3D rotation and translation from images only, i.e., the 3D origin of the camera ($\vec{o}_c$) cannot be determined from image correspondences. Nevertheless, most turntable motions are modelled as a circular motion because the perturbation of the screw axis is not significant.

3.4 Modified projection matrix for an approximate circular motion

Figure 3.4 illustrates the geometry of an approximate circular motion. Let I_0 be a reference image plane, $\hat{I}_i$ and I_i respectively represent the estimated and true image plane which have been rotated by θ_i degree from $\vec{o}_c$. Although the true camera centre $\vec{o}_i$ of I_i cannot be determined from 2D images only, the 2D projective homography H_{p} between $\hat{I}_i$ and I_i can be estimated when there are at least four pairs of correspondences. Therefore, a 3D-to- corresponding pair $\vec{x}^w \mapsto \vec{x}^i$ of I_i is related by

$$\vec{x}^i = H_{\mathrm{p}} K R_{\mathrm{int}} [R_z(\theta_i) \quad -\vec{o}_c] \vec{x}^w, \tag{3.19}$$

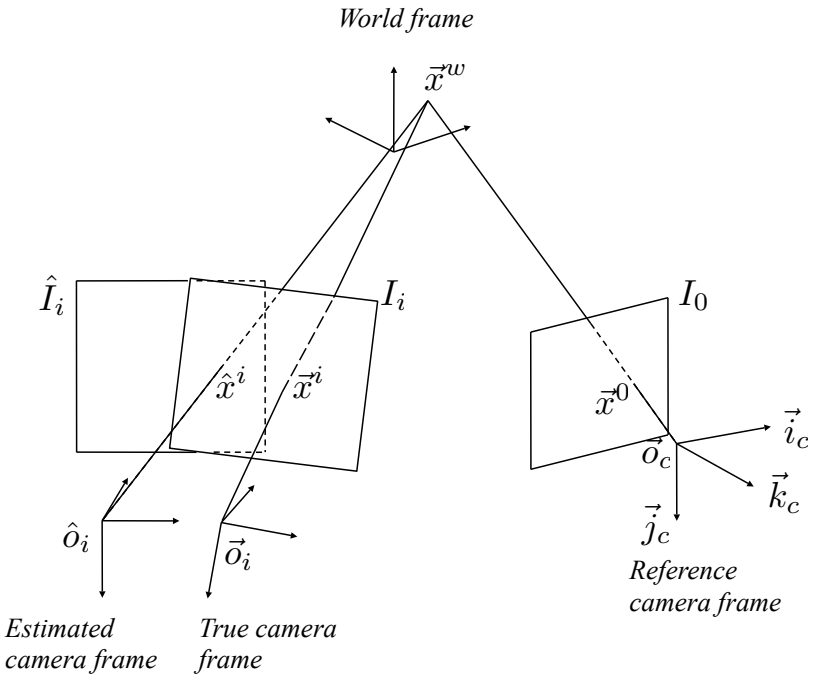


Figure 3.4: Geometrical illustration of a circular motion.

where H_{p} is designed as a projective homography because only projective transform in $\mathbb{P}^2$ can move a line at infinity, e.g., $\vec{l}_h$. A linear solution of H_{p} is derived from the DLT algorithm when more than four point correspondences are provided [3]. Therefore, the modified projection matrix for an approximate circular motion is

$$P'(\theta_z) = H'_{\mathrm{p}}[R(\theta_z) \quad -\vec{o}^c] \tag{3.20}$$

where $H'_{\mathrm{p}} = H_{\mathrm{p}} K R_{\mathrm{int}}$.

To locate true and approximate points in an image for the estimation of H_{p}, a method in [53] places four fiducial markers on the xy plane of the world frame (i.e., images of those points become images of true points in $\hat{I}_i$). New positions of the true points in a rotated view are located by the epipolar constraint and the proximity assumption, which premises a true point exists near to its estimation. Another method proposed in [19] assumes that an image of a string, which physically connects a rotation centre and a camera tripod, as an approximation of an image of the rotation axis. Thus, the initial

images are transformed to ensure that an image of a string should have fixed position in all views before estimating unknown rotation angle θ_z under a pure circular motion.

In this thesis, a wide-baseline image matching algorithm is exploited to search true points[5]. For example, the estimated points ($\hat{x}_i$) corresponding to the true points ($\vec{x}_i$) are located by the projection of true 3D points ($\vec{x}^w$), which are reconstructed by the linear triangulation from the first two views in an image triplet. Therefore, to reduce the triangulation error, two images of an image triplet should have reliable initial projection matrices. Since the rotation images near a reference position satisfy a pure circular motion well [see Figure 3.3(a)], projection matrices belonging to this region (called a trust region) are acceptable to use without modification. Thus, a projection modification algorithm increases the initial trust region until further modification is not required. This method is more practical than other two methods mentioned earlier since true 2D points for the modification are detected based on point correspondences between images. Chapter 4 introduces more details regarding a robust feature detection and matching strategies in two views.

3.5 Experimental results

To evaluate the performance of the proposed modification method, the following tests are conducted:

- comparing the re-projection error of Data set I with the error obtained after modification by the proposed method;
- visualisation of a conic error and deciphering the meaning in terms of an approximate circular motion;
- analysing the quality of VH results after modification.

The two graphs shown in Figure 3.5 illustrate the average re-projection errors of estimated projection matrices (marked as $\circ$) and modified projection matrices (marked as $\square$) relative to the rotation angle. The average error without the modification is 5.9981[px], which can be reduced to 0.5883[px] after modification of the projection matrices. The

[5]More details of the wide-baseline image matching is presented in Chapter 4

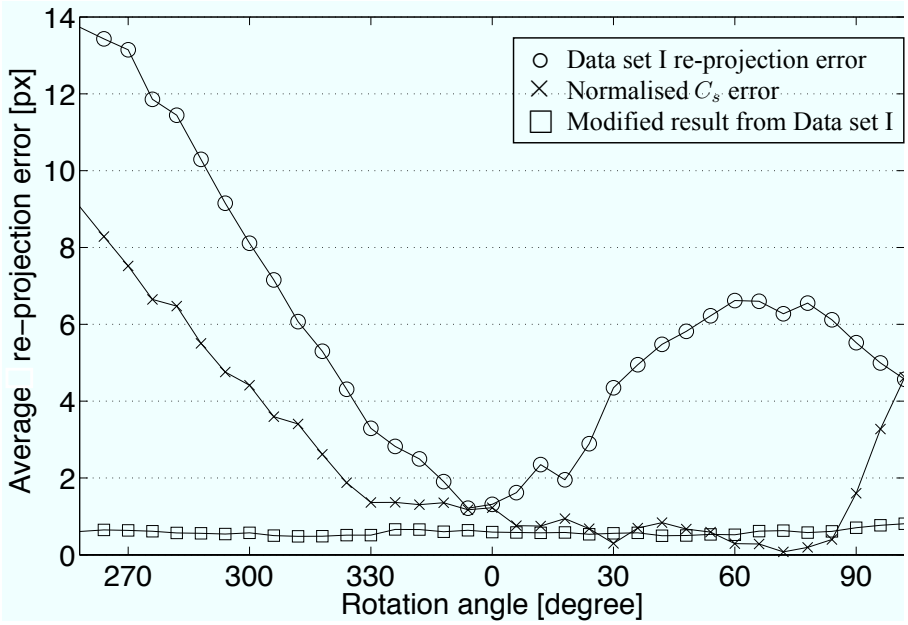


Figure 3.5: Average projection error before and after the projection modification.

graph marked $\times$ in the figure shows that the error between true and estimated line entities in a circular motion. To measure this error, a 3×3 matrix is defined from $\vec{l_s}$ and $\vec{l_h}$, i.e.,

$$C_{\mathrm{s}} = \vec{l_h}\vec{l_s}^{\mathrm{T}} + \vec{l_s}\vec{l_h}^{\mathrm{T}}. \tag{3.21}$$

The difference between the true and estimated C_{s} are measured by the largest eigen value of a difference of two matrices, i.e., it does not have a unit[6]. This error helps to infer the type of additional 3D motions involved in erroneous projection matrices.

Geometrically, a matrix C_{s} can be seen as a special form of a conic in $\mathbb{P}^2$. This is because it satisfies the condition of a conic [i.e., $(\vec{x}^m)^{\mathrm{T}} C_{\mathrm{s}} \vec{x}^m = 0$], but does not have full rank [i.e., $rank(C_{\mathrm{s}}) = 2$] unlike general conics [3]. Thus, a conic defined by two lines is often called a degenerate conic. A conic C_{s} always has a fixed shape if any of its two views are related by a pure circular motion (i.e., otherwise, the shape from an estimated conic is distorted). The conic differences around the reference position 0° are low (see a graph marked by $\times$ between $\pm 30°$ from the referece). However, the difference increases as the rotation goes to the left side (e.g., the maximum value 9.0615 is obtained at 258°).

[6]The difference shown in Figure 3.5 is normalised by the maximum C_s error for the visualisation.

Table 3.3: Seven-level volume reconstruction result before and after modification of projection matrices.

	After modification			Before modification[a]		
	Volume[b]	Points[c]	Octants	Volume	Points	Octants
Internal data	103.306	1568	618	99.1859	1541	613
Surface data	309.402	5793	3004	309.402	5749	3004
Total	412.708	7361	3622	408.588	7290	3617

[a]Seven-level of octree result obtained from images shown in Figure 3.7
[b]volume unit is cm^3
[c]the number of points after removing duplicated vertices

This large conic difference indicates that the estimation has an additional 3D translation of the camera centre, which violates the constraints of a pure circular motion. On the other hand, the conic differences from 0° to 84° are lower than the error at the reference position. This means that approximated two fixed line entities are almost identical to two true lines. Therefore, the higher average re-projection error in the right side is only explained by incorrect rotation measurement in the z axis.

For example, in an image at 258°, shown in Figure 3.6(a) where the dotted and solid lines respectively represent the true and estimation results, the difference between the true and estimated C_s is noticeable, i.e., the two fixed line entities do not coincide. This is because a hidden 3D translation is involved in this image, so that a true $\vec{x}_a$ does not lie on an image of true $\vec{l}_h$. On the other hand, the difference of two line entities in Figure 3.6(c) is not noticeable, and the locations of true ($\times$) and estimated 2D points ($+$) are almost identical near the rotation axis. However, the estimated positions become incorrect as the projection is further away from the rotating axis, and this error contributes large re-projection error. This mean the projection matrix of this image is modified by additional rotation in the z axis. Two images (b) and (d) illustrate modified image results, where the white boundary represents a true image plane in an approximated view.

The volumetric reconstruction results before and after modification of projection matrices are compared in Table 3.3. The total volume after modification is increased from 408.588 to 412.708 because the modification avoids incorrect status classification in a SfS technique. In particular, the volume of surface octants is unchanged but the

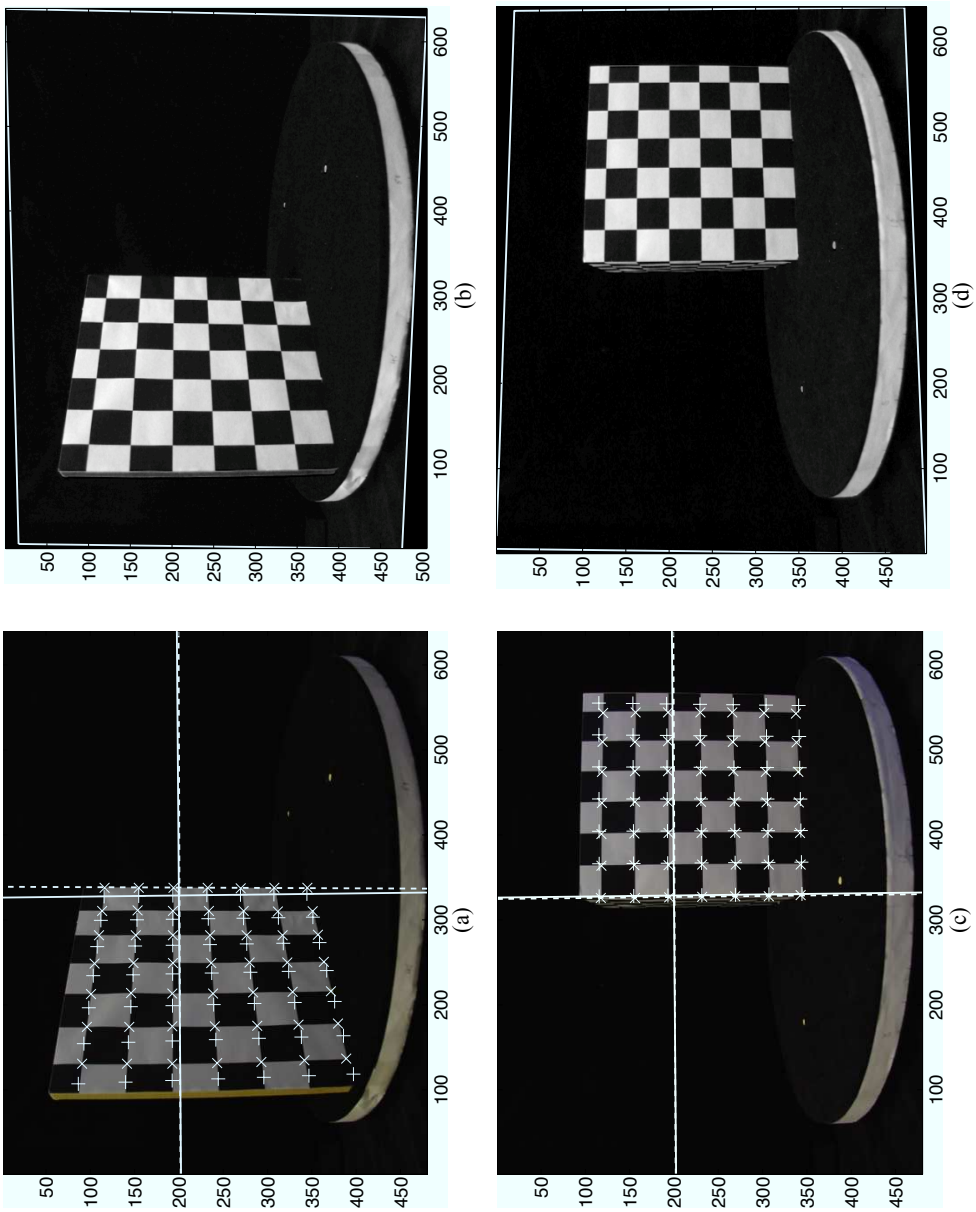


Figure 3.6: Example of two fixed lines in a pure circular motion: (a) the difference of true and approximated fixed line entities are noticeable due to additional 3D translation; (c) although two fixed lines entities are almost identical, the projection error is high for points further away from the rotation axis, i.e., additional rotation is involved in the z axis; (b) and (d) the modified image planes are denoted by white boundary.

Table 3.4: Eight-level volume reconstruction result before and after modification of projection matrices.

	After modification			Before modification[a]		
	Volume[b]	Points[c]	Octants	Volume	Points	Octants
Internal data	170.343	16171	6665	202.67	18119	7431
Surface data	129.618	33282	17397	139.56	36338	18731
Total	299.96	49453	24062	342.22	54457	26165

[a]Eight-level of octree result obtained from images shown in Figure 3.7
[b]volume unit is cm^3
[c]the number of points after removing duplicated vertices

volume of internal octants increases after the modification. This is because some of projection matrices are move their camera origins towards to the object (e.g., it is similar to zooming action) while other projection matrices remain unchanged. Nevertheless, total volume difference is not significant (1% increment). On the other hand, when the level of octree increases to eight, the difference becomes more noticeable (see Table 3.4). The modified calibration results confines the seven-level of octree result, which reduces the total volume from 342.22 to 299.96[cm^3]. For this reconstruction test, sixty turntable images are used in a seven-level of octree reconstruction. Some of input images which are transformed by the modified projective transforms are shown in Figure 3.7, where white boundary represents a true image plane.

3.6 Conclusions

An approximate circular motion involves hidden 3D motions which are not modelled in a pure circular motion. Although additional 3D motion in each image is not significant, the error caused by these motions propagates as a view is rotated away from the reference position. Therefore, images of an approximate circular motion should be modified in order to exploit useful constraints derived from a pure circular motion. As one approach to address this issue, this chapter proposes a method which exploits the fact that true and estimated image planes are related by a projective homography in $\mathbb{P}^2$. Therefore, the proposed modification process estimates an appropriate projective transforms from image

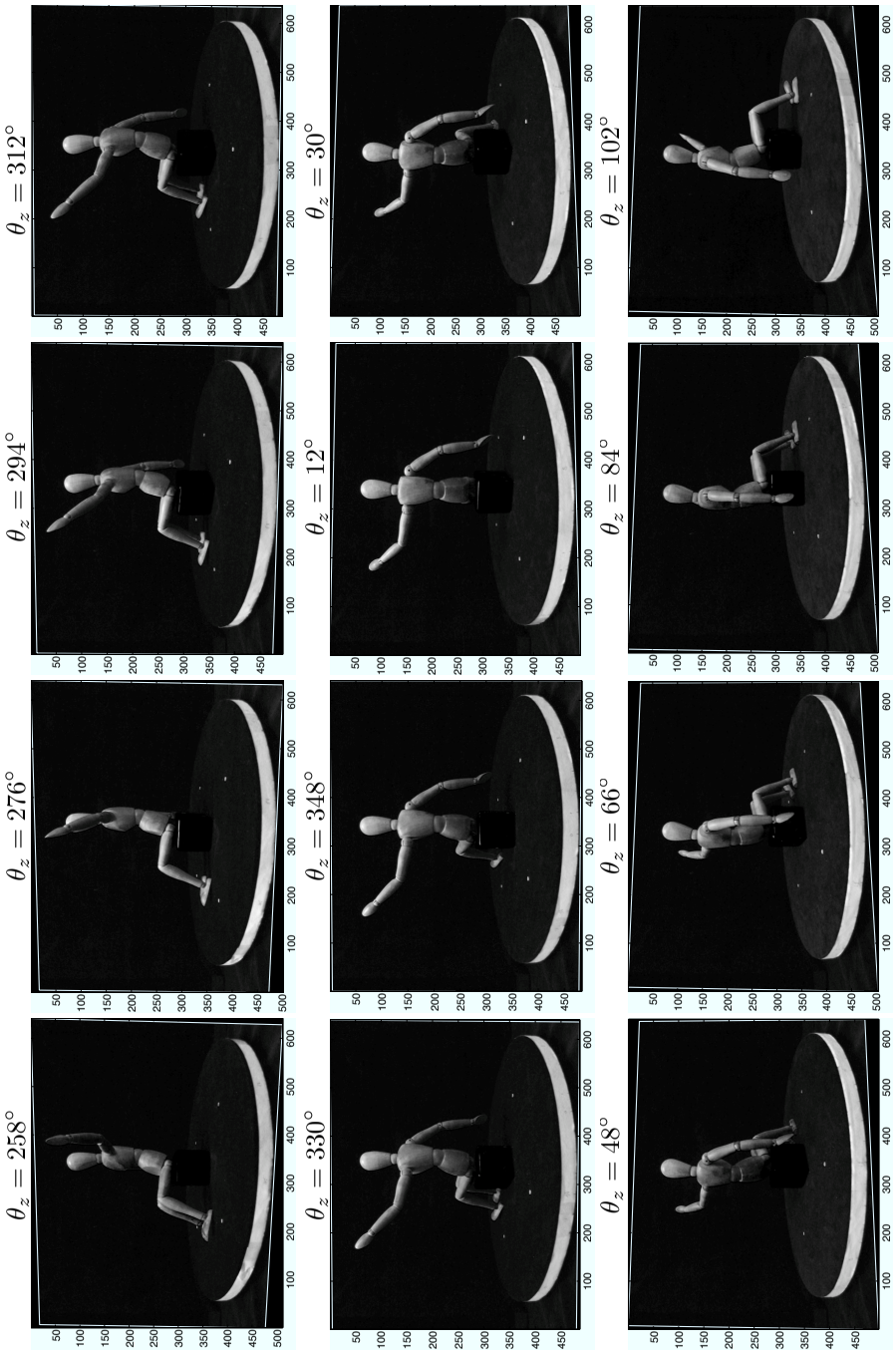


Figure 3.7: Example of modified images for the volumetric reconstruction by a SfS technique. White boundary in each image illustrates an image of a true image plane in an estimated view.

correspondences in an image triplet. In this way, this method provides the view of a true image plane in the approximated camera position. Thus, it is a different approach that directly searches a true projection matrix using an optimisation algorithm. Nevertheless, this modification is sufficient to determine the status of voxel correctly in a SfS method.

Chapter 4

Feature correspondences from wide baseline views

4.1 Introduction

Corresponding points between images give significant information for understanding a scene, which is particularly exploited for the visual hull refinement in this thesis. Before developing the refinement idea presented in the following chapter, this chapter introduces two matching strategies, which are inspired by a hypothesis that the use of grouped features can enhance the matching performance more than that of traditional point-based matching. Assuming point features are uniquely grouped in a Delauany graph, the first proposed method investigates geometric attributes (e.g., angles and edge lengths) extracted from a local point cluster, called a clique. Since these geometrical attributes are invariant to a similarity transform as the Delaunay graph is, the proposed can measure the distance between two cliques which are affected by rotation, translation and scaling. Furthermore, it inherits noise robustness from the Hausdorff distance (HD) [54] and has partial matching ability because the matching is performed on local entities.

The second method generalises the feature grouping concept to cope with affine transformed image matching. Instead of using geometric attribute of a clique, the second method exploits a feature descriptor, which is more distinctive and invariant to affine

transformation, and includes the local information near a feature point in a compact form. The second method proposes a new grouped feature descriptor called a clique descriptor, which is designed to involve neighbour descriptors as well as shape derived from the Delaunay graph of normalised point data. Experimental results show that the proposed clique descriptor matching methods produce more tentative correspondences than the previous methods.

This chapter is organised as follows. In Section 4.2, a Delaunay tessellation and its characteristics are explained. Also, this section introduces a construction algorithm and the theoretical representation of grouped points used in the proposed method. Section 4.3 explains the similarity invariant matching method, e.g., how the distance between two cliques is measured and how it is related to HD. Section 4.4 briefly explains the MSER detector which is used for an Invariant Region (IR) detector in proposed the affine invariant method. Normalised IR patch construction and existing descriptors [e.g., Scale Invariant Feature Transform (SIFT) and shape descriptor] are also explained in addition to the proposed clique descriptor and its matching method. Finally, experimental results and conclusions are presented in Section 4.5 and Section 4.6, respectively. In summary, this chapter presents explanation of following topics:

- a Delaunay graph construction used to cluster local features;
- a similarity invariant point matching method;
- an affine invariant point matching method.

4.2 Delaunay graph

A Delaunay graph is a result of Delaunay tessellation[1] and dual of a Voronoi diagram that divides distinct n points, called a site, according to the nearest neighbour rule, and the result forms a pattern of packed convex polygons [55, 56]. As a dual, the Delaunay graph is constructed from connecting sites of Voronoi polygons that are adjacent to one another [57]. For example, Figure 4.1(a) and (b) illustrate a Voronoi diagram and its dual graph (i.e., Dealuany graph) from 20 randomly generated 2D points marked as *. A

[1] In Chapter 6, it is also referred to as Delaunay Triangulation (DT) due to the triangular face of the simplex of a 2D Delaunay graph.

triangle becomes a simplex of the graph in the 2D case and the number of points defining a simplex increases as the dimension is increased, e.g., a tetrahedron replaces a triangle in a 3D space. The fundamental condition of creating a simplex is that it does not allow any point within the circle circumscribing a simplex. Suppose that an initial guess constructs a simplex from three points, $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$, and there is another point $\vec{v}_4$. The point $\vec{v}_4$ needs to be checked whether it conflicts with the circle defined by the initial simplex. In other words, if the following decision function is positive [58]

$$f_{del}(\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4) = \begin{vmatrix} x_1 & y_1 & (x_1^2 + y_1^2) & 1 \\ x_2 & y_2 & (x_2^2 + y_2^2) & 1 \\ x_3 & y_3 & (x_3^2 + y_3^2) & 1 \\ x_4 & y_4 & (x_4^2 + y_4^2) & 1 \end{vmatrix}, \tag{4.1}$$

where $\vec{v}_i = [x_i \ y_i]^T$ and $|\cdot|$ is the determinant of a matrix, then the initial triangle is considered correct, otherwise a new configuration of three points is generated. This means noise only affects a local graph where the circumcircles are contaminated [see Figure 4.1(c)]. Furthermore, even when points are partly occluded, the resulting graph is similar to that of the original graph [see Figure 4.1(d)].

Mathematically, the representation of a Delaunay graph is similar to that of an ordinary graph. Suppose that there are point sets of a model $\mathcal{V}^{\mathrm{m}} = \{\vec{v}_1, \cdots, \vec{v}_{|\mathcal{V}^{\mathrm{m}}|}\}$, where $|\cdot|$ is cardinality of a point set and $\mathcal{V}^{\mathrm{t}}$ is a test set. A Delaunay graph is then defined by sets of points, edges and faces, i.e., a graph $\mathcal{G}^{\mathrm{m}} = (\mathcal{V}^{\mathrm{m}}, \mathcal{E}^{\mathrm{m}}, \mathcal{F}^{\mathrm{m}})$, where $\mathcal{E}^{\mathrm{m}}$ is the list of node connections (i.e., $\mathcal{E}^{\mathrm{m}} = \{(i,j) \mid \forall \vec{v}_i, \ \vec{v}_j \in \mathcal{V}^{\mathrm{m}}\}$), and $\mathcal{F}^{\mathrm{m}}$ is a set of triplets of indices, in which each triplet represents a triangle such as $\mathcal{F}^{\mathrm{m}} = \{(i,j,k) \mid \forall (i,j), (j,k)$ and $(j,k) \in \mathcal{E}^{\mathrm{m}}\}$. A local entity of a graph called a clique is then defined as a cluster of points connected by $\mathcal{E}^{\mathrm{m}}$. By the clique notation in [59], a model clique centred at a point $\vec{v}_i$ is given by

$$\mathcal{C}_i^{\mathrm{m}} = \{i\} \cup \{j | \forall (i,j) \in \mathcal{E}^{\mathrm{m}}\} \ . \tag{4.2}$$

A centre point $\vec{v}_i$ of a clique $\mathcal{C}_i^{\mathrm{m}}$ is referred to as the seed of a clique and the indices of the other points are called neighbours, which are ordered in the clockwise direction. Some examples of a clique are illustrated in Figure 4.2, where $\vec{v}_{m,n}$ denotes a the n-th

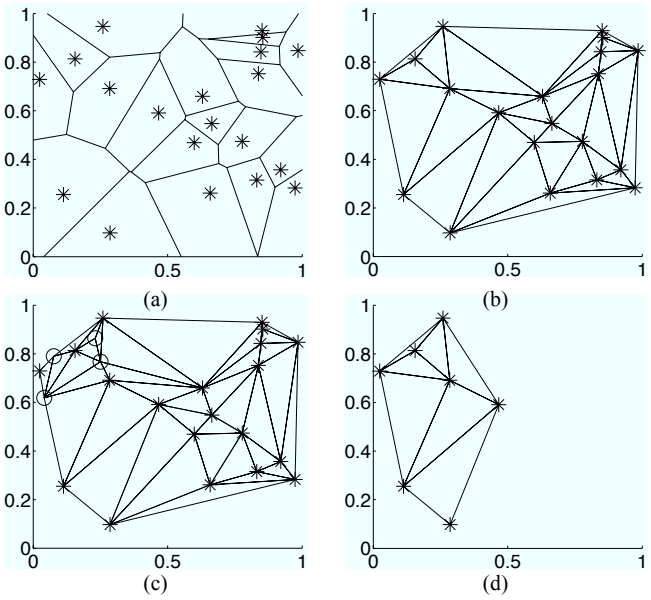


Figure 4.1: (a) A Voronoi diagram of 20 feature points (denoted by *) that are randomly generated and ranged [0 1]. (b) A Delaunay graph, the dual of the Voronoi diagram in (a). (c) The general shape of Delaunay graph in (b) is not changed by the addition of a noise point o. (d) A randomly selected small portion of features does not change the local graph significantly.

neighbour point of a clique with a seed point $\vec{v}_m$.

One benefit of using these structured points is that it creates a unique local configuration of points which is not as complicated as a generic non-directional graphical model, e.g., Markov random fields [60]. This is because the Delauany graph restricts the number of point connections as a triangle and the connections are only made within the nearest neighbours. However, a problem occurs when comparing two cliques having different size, i.e., $|\mathcal{C}_i^{\mathrm{m}}| \neq |\mathcal{C}_j^{\mathrm{t}}|$ as illustrated in Figure 4.2 in which the size of a model clique in Figure 4.2(a) is 7 and the size of the matching opponents in Figure 4.2(b) is 4. Thus, if a general point-pair based distance is used for measuring the difference of two cliques, it is necessary to establish inexact matching of neighbour points in advance, e.g., some points should have multiple opponents, or a null point concept is required [61, 62]. However, the proposed matching framework enables distance of two data sets with different sizes to be measured.

The traditional Delaunay construction algorithm shown in [55] updates a triangle whenever a new point is inserted into the current graph, i.e., all simplexes (i.e., triangles in

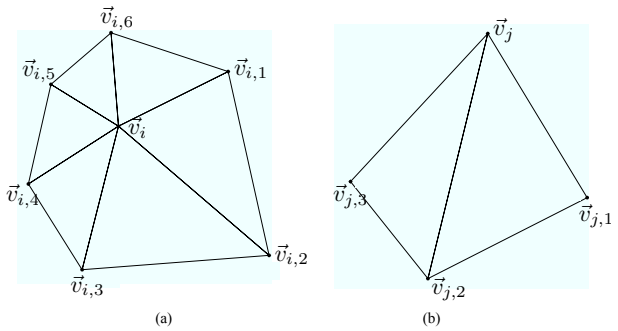


Figure 4.2: Example of a clique: (a) a model clique C_i^m; (b) a test clique C_j^t, where $\vec{v}_i$ and $\vec{v}_j$ are the seeds of the cliques.

```
1  // 1. Estimate a super triangle which tightly encloses all points in V
2  // 2. Add three points from the super triangle to V
3  // 3. Initialise a triangle table T with the super triangle
4  For i = 1 until the end of V
5     Add random offset to the i-th point v(i) in V
6     initialise convex list C
7     for j = 1 until the end of T
8        compute (4.1) from t(j,1), t(j,2), t(j,3), and v(i)
9        and store the result in d
10       if d > 0
11          delete the j-th triangle in T
12          and store its vertices in C
13       end if
14    end for
15    build new triangle which connects v(i) to points in C
16 end for
17 remove all triangles in T which connected to any vertex of a super triangle
18 remove three vertices of a super triangle from V
```

Figure 4.3: Pseudo code for a 2D Delaunay graph construction.

a 2D space) in the table are tested to determine whether the point violates the condition of (4.1). Thus, the complexity of this algorithm is $O(n^2)$ in 2D case, and $O(n^{\lceil \frac{d}{2} \rceil + 1})$ in d dimensional space [63]. A pseudo code for the Delaunay graph construction is shown in Figure 4.3, where V and T denote a vertex list and triangle list, respectively and v(i), t(j,k) represent the i-th point in V and the the k-th point in the j-th triangle of T. The algorithm first determines a super triangle that encloses all points in V and it is temporarily added to T as well as its three vertices which are stored to T (see line 1-3). This prevents the length of a boundary of a Voronoi diagram becoming indefinite. Thus, any triangles which are connected to the super triangle should be removed after completing estimation (line 17). There are other algorithms [64] that are computationally beneficial, e.g., an algorithm proposed in [63] stores the intermediate triangle record in a tree to facilitate the search process, which reduces the complexity of d dimensional Delaunay graph construction to $O(n^{\lceil \frac{d}{2} \rceil})$ and $O(n \ln n)$ in 2D case.

Some point configurations can fail to construct a simplex, e.g., coincide points,

collinear points (or coplanar points in 3D case), and four or more points in cyclic, i.e., four territories meet at a point [55]. Although input points are not fallen into those categories, round off error may regard two geometrically close points as an identical point. To avoid theses degeneracies, random offset is intentionally added to a point in line 5, even though it produces a sliver triangle at a cost of the stability of the algorithm. Alternatively, some methods simply ignored degenerate points.

4.3 Similarity invariant graph matching

4.3.1 Point pattern matching

Point Pattern Matching (PPM) is a problem that searches the best point correspondences by investigating underlying point pattern and it has been included in many vision applications, e.g., motion estimation, image registration and object recognition [60, 61, 65, 66]. Although much research has been intensely done to address this problem, the PPM result is not reliable particularly when a point set includes noise or outliers, when whole or part of the data has been transformed (e.g., rotated, translated and scaled) and when the number of matching points is different, i.e., some points should have either multiple matching opponents or none.

One classical approach to PPM is a spectral method that compares the correlation of eigen vectors of a distance matrix referred to as a proximity matrix. The intra-distances between all possible pairs of points in an image are measured by a Gaussian-weighted distance metric and stored as a matrix in which the eigen vectors are extracted as new features for matching [66]. Although the earlier methods show a weakness in dealing with partial matching, noise and significant image transformation, more recent methods address partial matching using a sub-matrix matching algorithm [67] and achieve robustness against noise by combining the spectral analysis with the Expectation Maximization (EM) framework [68]. Caelli et al. improve the accuracy of matching by re-normalising the eigen vectors and values used in comparison. The similarity of graph or tree of vertices is then measured by the distance of clusters in a tree or graph in the re-normalised subspace instead of the general point distance [69].

Another approach is HD matching. The key advantage of using HD is that exact

matching is not required when computing distance of two point clusters. Without a knowledge of exact point correspondences, HD measures distance between two sets of points. Furthermore, it allows small perturbation and partial point pattern matching [54]. HD is a non-directional and non-linear operation, and by making some changes in the distance function and combining two directional HD's, useful variants of HD can be generated, e.g., ranked HD and modified HD (see more details Appendix D) [70]. These methods have been extended to line and curve features in accurate recognition systems [71, 72, 73]. However, the matching performance of the traditional HD deteriorates when points are transformed. A recent algorithm which associates an affine invariant coordinate called homogeneous barycentric coordinates with HD, has been introduced to address the affine transformed point pattern matching [74], but barycentric coordinates rely on the convex shape of points and its mean position. Therefore, if the convex shape is deformed by an outlier, the matching result is not reliable.

Since the Euclidean distance of point features is not always sufficient for matching, some methods utilise contextual conditions derived in the point pattern. These methods structure points as a tree or a graph, which is used to solve the general PPM problems. In particular, weighted connection of nodes in a graph indicates the strength of connection and this concept is utilised by stochastic approaches, e.g., probabilitic relaxation methods [75]. Li et al. introduced a tree structure which is obtained by applying the k-D tree algorithm to partition points, and the similarity of trees is measured for sparsely distributed point patterns [76]. Strickland et al. suggested a method that uses an iterative relaxation algorithm based on the probability defined by a separation of length and angle between connected branches in a non-rigid shape [65]. Zheng et al. considered point matching as an optimisation problem in order to preserve local neighbourhood structure, and the optimal solution is searched using relaxation labelling [61]. Andrew et al. focussed on the Delaunay graph of a set of data points and suggested a relaxation labelling solution for Delaunay graph matching which simplifies the joint probabilities defined by neighbouring nodes and reduces the computation time of the iteration involved [59]. The concept is extended to a transform estimation in an EM framework in their later work [62]. Despite these extensive researches, PPM still remains a challenging task: spectral methods are not robust to corruption in structure (e.g., due to noise) and the performance of relax-

ation methods degrade when there are significant increases in the sizes of the point sets since defining every joint probability is complicated [60].

In this section a Delaunay graph is investigated as a method for uniquely constructing a local configuration, and propose a clique-based HD which exploits a geometrical shape difference between cliques. In the proposed method, since an idea of robust matching is motivated by HD, it effectively addresses noise, outliers, occlusion and inexact graph matching. Also, the similarity invariant features from the cross ratio can cope with the spatial transformation of points. The proposed distance measures inter-distance, i.e., it measures a clique distance directly from a model to a test data unlike a spectral method. Thus, when the sizes of the two graphs being compared are different, a rectangular matrix stores all the possible distances between cliques and the strong correspondences are identified to define the initial transform between two sets. Once the local transform is estimated initially, the guided matching increases point correspondences by collecting point pairs (called supporting pairs) that reside within an error bound of the estimated transform and repeats the estimation.

4.3.2 Clique distance

A similarity transform H_{s} has 4 DoF as explained in Appendix B, i.e., H_{s} describes a rigid motion of an object with a change in scale in a 2D space. Some of its useful invariant properties of $H_{mathrms}$ are that length ratio, area ratio and the combination of vectors (e.g., centroid) are preserved under the similairty transformation [3]. Therefore, the Delaunay graph of points transfomred by H_{s} also remains unchanged. This invariance is exploited in a new distance measure between cliques.

Instead of a direct use of edge length and angles extracted from a clique, the proposed geometrical distance compares a set of cross ratios defined on boundary edges of a clique. The cross ratio is the length ratio of four distinct points on a collinear line, i.e., the cross ratio of four points, $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$, and $\vec{v}_4$ is

$$f_{\mathrm{cr}}(\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4) = \frac{||\vec{v}_1 - \vec{v}_3|| \cdot ||\vec{v}_2 - \vec{v}_4||}{||\vec{v}_1 - \vec{v}_4|| \cdot ||\vec{v}_2 - \vec{v}_3||} \ . \tag{4.3}$$

Since the cross ratio is invariant up to the projective transform [1], it can correctly

measure the shape difference of two similarity transformed graphs. To obtain four points for the cross ratio estimation, the proposed method exploits the midpoints of two sides of a triangle belonging to a clique, and the two ends of the boundary edge. Since these four points are not collinear, the two midpoints are projected onto the boundary line. Suppose a neighbour index of $\mathcal{C}_i^{\mathrm{m}}$ is denoted as $n_{i1}, \cdots, n_{i(|C_i^{\mathrm{m}}|-1)}$. The boundary of a clique $\mathcal{C}_i^{\mathrm{m}}$ is then defined as

$$\mathcal{B}_i^{\mathrm{m}} = \{(n_{ij}, n_{i(j+1)}) | j = 1, \cdots, |\mathcal{C}_i^{\mathrm{m}}| - 2\} \ . \tag{4.4}$$

If the seed of a clique is enclosed by its neighbours [see Figure 4.2(a)], the boundary set has another element $\{(n_{i(|\mathcal{C}_i^{\mathrm{m}}|-1)}, n_{i1})\}$. This is because the face of an internal clique is normally defined by a circular permutation of two adjacent neighbours but a clique whose boundary is identical to the boundary of a graph is not, i.e., the maximum size of a boundary set is limited by $|\mathcal{B}_i^{\mathrm{m}}| \leq |\mathcal{C}_i^{\mathrm{m}}| - 1$.

All the midpoints from two side edges of a triangle in $\mathcal{C}_i^{\mathrm{m}}$ are stored in

$$M_i^{\mathrm{m}} = [[\vec{\mu}_{i1}^1 \vec{\mu}_{i1}^2][\vec{\mu}_{i2}^1 \vec{\mu}_{i2}^2] \cdots [\vec{\mu}_{i|\mathcal{B}_i^{\mathrm{m}}|}^1 \vec{\mu}_{i|\mathcal{B}_i^{\mathrm{m}}|}^2]] \ , \tag{4.5}$$

where $\vec{\mu}_{ij}^1 = 0.5(\vec{v}_i + \vec{v}_{n_{ij}})$ and $\vec{\mu}_{ij}^2 = 0.5(\vec{v}_i + \vec{v}_{n_{i(j+1)}})$, i.e., $\vec{\mu}_{ij}^2 = \vec{\mu}_{i(j+1)}^1$. Consequently, each triangle in a clique is represented by the cross ratio, i.e. a triangle of $(i, n_{ij}, n_{i(j+1)})$ is defined by

$$r_{ij} = f_{\mathrm{cr}}\left(f_{\mathrm{o}}(\vec{v}_{n_{ij}}, \vec{\mu}_{ij}^{1'}, \vec{\mu}_{ij}^{2'}, \vec{v}_{n_{i(j+1)}})\right) \ , \tag{4.6}$$

where $\vec{\mu}_{ij}^{k'}$ is a projection point of $\vec{\mu}_{ij}^k$ onto the boundary vector $\vec{v}_{n_{i(j+1)}} - \vec{v}_{n_{ij}}$, and $f_{\mathrm{o}}(\cdot)$ represents an ordering function which orders four points in the direction of the boundary vector. Figure 4.4 illustrates some examples of geometrical distances of a Delaunay graph from 20 random points. When a triangular face has an obtuse angle, the projections of midpoints lie outside of the triangle [see Figure 4.4(b)]. In this case, the order of four points is different from the order of the acute triangle shown in Figure 4.4(a). To make the order of points consistent, an ordering function $f_{\mathrm{o}}(\cdot)$ rearranges the four points in the direction of the boundary vector and any direction of the boundary gives the same cross ratio, i.e., $f_{\mathrm{cr}}(\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4)$ is equal to $f_{\mathrm{cr}}(\vec{v}_4, \vec{v}_3, \vec{v}_2, \vec{v}_1)$. Thus, the face $\mathcal{F}^{\mathrm{m}}$ can also

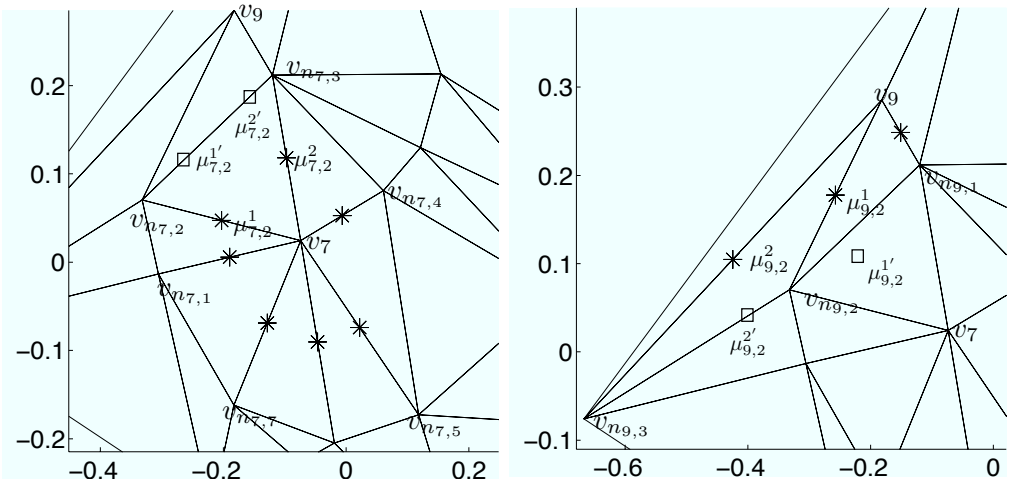


Figure 4.4: Illustration of a geometrical distance: (a) a triangle defined by $(\vec{v}_7, \vec{v}_{n_{7,2}}, \vec{v}_{n_{7,3}})$ is a face of $\mathcal{C}_7$ and its cross ratio is $f_{\text{cr}}(\vec{v}_{n_{7,2}}, \vec{\mu}^{1'}_{7,2}, \vec{\mu}^{2'}_{7,2}, \vec{v}_{n_{7,3}})$; (b) for a face $(\vec{v}_9, \vec{v}_{n_{9,2}}, \vec{v}_{n_{9,3}})$ with an obtuse angle, the projection of a midpoint is outside of its boundary.

be represented in terms of the cross ratios of $\mathcal{C}^{\text{m}}_i$, i.e., $\mathcal{F}^{\text{m}} = \{R^m_1, \cdots, R^m_{|V^m|}\}$, where $R^m_i = [r_{i1} \ \cdots \ r_{i(|\mathcal{B}^{\text{m}}_i|)}]$.

The geometrical distance between two cliques is incorporated in the modified HD, where the size of the triangle is used for the weight, i.e.,

$$h_{\text{g}}(R^{\text{m}}_k, R^{\text{t}}_l) = \frac{1}{|R^{\text{m}}_k|} \sum_{\alpha \in R^{\text{m}}_k} \min_{\beta \in R^{\text{t}}_l} \{g(r_{k\alpha}, r_{l\beta})\} \ , \tag{4.7}$$

where $g(\cdot)$ is

$$g(r_{k\alpha}, r_{l\beta}) = (1 + ||\frac{a_\alpha}{a_{\mathcal{C}^{\text{m}}_k}} - \frac{a_\beta}{a_{\mathcal{C}^{\text{t}}_l}}||)||r_{k\alpha} - r_{l\beta}|| \ , \tag{4.8}$$

and $a_{\mathcal{C}^{\text{m}}_k}$ is the total area of a clique $\mathcal{C}^{\text{m}}_k$, and a_α represents the area of a triangle indexed by α. Thus, the geometrical distance accounts for the difference of area ratio in a clique. The non-directional version of the geometrical distance $h_2(R^{\text{m}}_k, R^{\text{t}}_l)$ is obtained by choosing the maximum of two directional distances. Finally, a geometrical proximity matrix is defined as

$$\Delta_{\text{g}} = \begin{bmatrix} h_2(R^{\text{m}}_1, R^{\text{t}}_1) & h_2(R^{\text{m}}_1, R^{\text{t}}_2) & \cdots & h_2(R^{\text{m}}_1, R^{\text{t}}_{|\mathcal{V}^{\text{t}}|}) \\ \vdots & \vdots & \cdots & \vdots \\ h_2(R^{\text{m}}_{|\mathcal{V}^{\text{m}}|}, C^t_1) & h_2(R^{\text{m}}_{|\mathcal{V}^{\text{m}}|}, R^{\text{t}}_2) & \cdots & h_2(R^{\text{m}}_{|\mathcal{V}^{\text{m}}|}, R^{\text{t}}_{|\mathcal{V}^{\text{t}}|}) \end{bmatrix} \ , \tag{4.9}$$

and the matrix is normalised by the maximum value of its elements. A clique distance matching can cope with partial occlusion, some noise level and similarity transformation, but point distribution is not sufficient to recognising an object in practical case, i.e., a more distinctive feature should accompany the feature grouping approach. In this context a feature descriptor is explored and incorporated in the second matching method.

4.3.3 Guided matching

If one global transform H_{s} is assumed between point patterns, then it is given by minimising

$$\arg\min_{H_{\mathrm{s}}} \left(\sum_{\alpha=1}^{|\mathcal{V}^m|} ||H_{\mathrm{s}}[\vec{v}_\alpha \ 1]^{\mathrm{T}} - [\vec{v}_\beta \ 1]^{\mathrm{T}}|| \right), \tag{4.10}$$

where $\vec{v}_\beta$ is a matching opponent to $\vec{v}_\alpha$ in $\mathcal{V}^t$, i.e., $\vec{v}_\alpha \leftrightarrow \vec{v}_\beta$. Suppose that only a subset of the test data perfectly matches the model in terms of geometrical distance due to significant noise contamination and partial occlusion. There is then a possibility to obtain more corresponding pairs by means of an initial transform estimated by (4.10) from those perfect matches, and this process is called a guided matching. From the geometrical proximity matrix in (4.9), a set of strong correspondences are selected if the distance is smaller than the local error ϵ_l, i.e.,

$$\hat{L} = \{\vec{v}_k \leftrightarrow \vec{v}_l \mid \Delta_{\mathrm{g}}(k,l) \leq \epsilon_l\}. \tag{4.11}$$

A closed form solution of H_{s} is derived by revising (4.10) using $\hat{L}$. For example, if there are strong correspondences $\vec{v}_i = [x_i \ y_i] \leftrightarrow \vec{v}'_i = [x'_i \ y'_i]$ in $\hat{L}$, then the initial transform $\hat{H}_{\mathrm{s}}$ should satisfy the condition, $\hat{H}_{\mathrm{s}}[x_i \ y_i \ 1]^{\mathrm{T}} - [x'_i \ y'_i \ 1]^{\mathrm{T}} \simeq 0$. This relation is re-expressed in terms of similarity parameters ϕ_1, ϕ_2, t_x, t_y, i.e.,

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & -x'_1 \\ y_1 & -x_1 & 0 & 1 & -y'_1 \\ & & \vdots & & \\ x_i & y_i & 1 & 0 & -x'_i \\ y_i & -x_i & 0 & 1 & -y'_i \end{bmatrix} \begin{bmatrix} \phi_1 & \phi_2 & t_x & t_y & 1 \end{bmatrix}^T = \vec{0}, \tag{4.12}$$

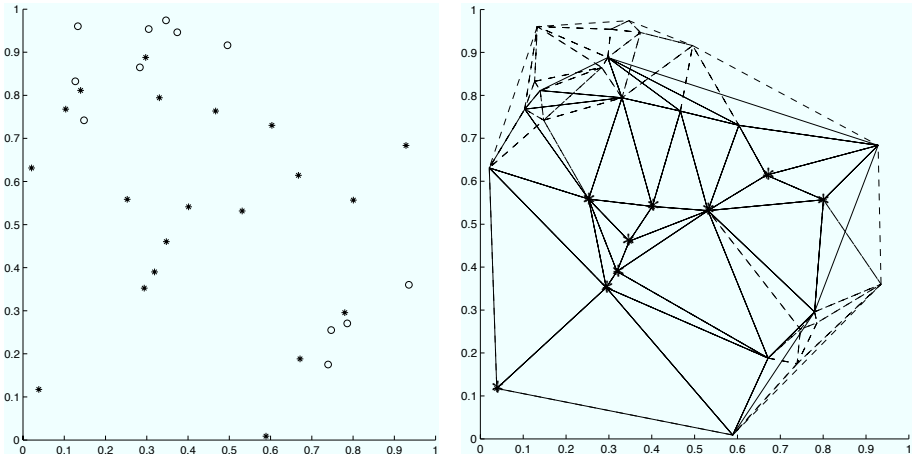


Figure 4.5: (a) Noise points (denoted by o) and signal points (denoted by *). (b) Solid line represents the Delaunay graph of the signal points only; dashed graph represents the Delaunay graph due to the noise points; and strong correspondences are denoted by *.

where $\phi_1 = s\cos\theta$ and $\phi_2 = s\sin\theta$. Thus, the linear solution exists in the null space of the matrix in (4.12), and four similarity parameters can be estimated when $|\hat{L}| \geq 2$ ideally, but a stable result is expected when $|\hat{L}| \geq 4$.

Examples of strong correspondences of signal points contaminated with noise are shown in Figure 4.5. Although most cliques in the graph have additional faces due to the noise, some parts of the graph remain unchanged. Correspondences with geometrical distance less than ϵ_l are selected as strong correspondences [see * in Figure 4.5(b)], which are used for the estimation of an initial transform, $\hat{H}_{\mathrm{s}}$. A set of guided matching pairs L is then given by

$$L = \hat{L} \cup \{\vec{v}_k \leftrightarrow \vec{v}_l | |\hat{H}_{\mathrm{s}}[\vec{v}_k\ 1]^{\mathrm{T}} - [\vec{v}_l\ 1]^{\mathrm{T}}|| \leq \epsilon_r\}, \tag{4.13}$$

where ϵ_r is the estimation threshold, and a refined transform H_{s} is estimated from L. The above procedure (i.e., correspondence improvement and transform refinement) are applied iteratively, and if a significant number of points satisfy the result of the n-th iteration then the estimated local transform is considered as a global transform, and points with values greater than the threshold are regarded as noise. The guided matching process is similar to the random sample consensus algorithm which expects strong correspondences to be randomly sought (see more details about the random sample consensus algorithm

in Appendix E).

4.4 Clique descriptor matching

4.4.1 Feature descriptor matching

A feature descriptor is generally designed to describe the local photometric information around a feature and it enhances matching performance significantly when an image is deformed by noise, illumination change, affine transform, scale change and 3D view point change. For example, a descriptor proposed in SIFT is one of the most successful feature descriptor, where a histogram of locally reoriented image gradient is used to describe the local characteristics of a feature [77]. Mikolajczyk et al. claims that SIFT-based descriptors, e.g., SIFT, SIFT-PCA and GLOH, perform best amongst other state-of-the-art descriptors [78]. However, since the feature points used in SIFT are defined on a scale invariant region not on an affine invariant region, error is inevitable when matching image which is affine transformed. Harris affine and Hessian affine detectors try to modify scale invariant region in order to adapt affine invariance iteratively using the fact that corresponding normalised affine regions have similarity up to 2D rotation.

Other researches for finding affine invariant regions are motivated by wide baseline image matching where Invariant Regions (IR's) are used instead of point features for matching because the projection of plane-like surface is locally well modelled by an affine transform [79]. Tuytelaars et al. proposed two methods for IR detection. One uses corner and nearby edges, and the other uses the intensity function along rays emanating from the local intensity extremum to estimate an elliptical IR [80]. A Maximally Stable Extremal Region (MSER) detector proposed by Matas et al. is also an intensity-based IR detector from single image and it has highly desirable properties, e.g., extremal regions are closed under continuous geometric transform and monotonic transform of image intensity [81]. The MSER detector has been extended to detect maximally stable colour regions [82].

The most intuitive IR matching scenario is to compare image correlations of IR's and establish tentative correspondences from highly correlated IR's followed by a more robust matching algorithms (e.g., RANSAC) which exploits epipolar constraint of two views [3, 83]. Alternatively, the Mahalanobis distance between IR's or scaled IR's can

be used to increase tentative correspondences rather than a simple correlation measure [81]. However, for more robust matching, it is better to use a distinctive descriptor of a normalised IR patch instead of a direct use of the texture of an IR. In particular, a rotation invariant IR descriptor is preferred because two corresponding normalised IR's have similarity up to 2D rotation. Schaffalitzky et al. proposed a texture region descriptor where a rotationally invariant bank of local operators represents texture regions extracted from an over-segmentated image [84]. Lowe applied his SIFT descriptor to a MSER with χ^2 distance [85] because SIFT descriptor uses local gradients which are reoriented by locally dominant gradients. Chum et al. also proposed a nontexture-based IR descriptor defined by two local affine frames and this is used for indexing a geometric hash table in order to perform IR matching in constant time [86].

The latest endeavour to increase tentative correspondences in descriptor matching uses the local neighbours of an IR. As a spatial IR proximity, the k-th nearest neighbour is used in Lowe's pair descriptor and pair matching distance decides correspondences. Thus, whenever a match is found, two pairs of IR's are added to tentative correspondences [85]. The performance of this approach is normally similar to SIFT descriptor matching but is better in a scene with near occlusion. In this chapter the pair descriptor concept is extended to a group descriptor referred to as a clique descriptor. A clique descriptor consists of a seed IR descriptor and neighbour descriptors in the Voronoi space of normalised points. To ensure the robustness of a clique descriptor matching to noise, a modified Hausdorff distance is adopted for neighbour distance, and the seed and neighbour distance are appropriately weighted. This matching method increases tentative correspondences between initial images and a new image taken from a different camera position.

4.4.2 MSER detector

The MSER detector is used in the proposed matching method to detect IR's due to its simplicity and fast implementation - basically, it easily detects IR's from subsequent image thresholding. A MSER is defined solely by an extremal property of the intensity function in the region and its outer boundary [81]. Let $I(\vec{p})$, where $\vec{p}$ is a position vector at image, be a function that returns intensity values of a set $\mathcal{I}$, e.g., 8 bit grey level image

has $\mathcal{I} = \{0, 1, 2 \cdots, 255\}$. A maximum intensity region $\mathcal{R}_m$ is then defined by

$$\mathcal{R}_m = \{\vec{p} \mid I(\vec{p}) > I(\vec{q}), \text{where } \forall \vec{p} \in \mathcal{R}, \forall \vec{q} \in \partial\mathcal{R}\} \ , \tag{4.14}$$

where $\mathcal{R}$ represents a region in an image, i.e., a set of 8-connected neighbour points and $\partial\mathcal{R}$ is its boundary. Thus, the minimal intensity region $\mathcal{R}_n$ is defined by the opposite condition of $\mathcal{R}_m$, i.e., $I(\vec{p}) < I(\vec{q})$. The MSER detector determines IR's from every sequence of nested extremal regions that satisfy the stability condition

$$d(\mathcal{E}_i) = \frac{|\mathcal{E}_{i+\Delta}| - |\mathcal{E}_{i-\Delta}|}{|\mathcal{E}_i|} \ , \tag{4.15}$$

where $|\cdot|$ denotes the number of elements in a set and Δ is a small increment. For example, suppose that there is a sequence of nested extremal regions, $\mathcal{E}_1 \subset \mathcal{E}_2 \cdots \subset \mathcal{E}_k$. The i-th extremal region $\mathcal{E}_i$, where $(1 < i < k)$ is then selected as a maximally stable extremal region when $\mathcal{E}_i$ is a local minimum of (4.15). Therefore, a MSER comprises binarised regions whose areas do not change significantly in adjacent thresholding values.

After IR detection, the covariance matrix of a MSER defines an elliptical IR, i.e., a maximally stable $\mathcal{E}_i$ is represented by a 2×2 matrix, $C_i = |\mathcal{E}_i|^{-1} \sum_{\vec{p} \in \mathcal{E}_i} (\vec{p} - \vec{m}_i)(\vec{p} - \vec{m}_i)^{\mathrm{T}}$, where $\vec{m}_i$ is the mean of $\mathcal{E}_i$ and the quantity of anisotropy is measured by the ratio of two eigen values of C_i. The eigen vectors of C_i and $\vec{m}_i$ define a local reference frame of an IR which is used later for searching a local neighbourhood.

A MSER normalisation is a process to transform various elliptical IR's of different orientations and scales to $N_p \times N_p$ square image patches for robust matching. This process is similar to a random signal whitening process that transforms a random data with a high anisotropy ratio to a new data with a normalised covariance matrix. A covariance matrix can be decomposed to $C_i = U diag(\lambda_1, \lambda_2) U^{\mathrm{t}}$, where $UU^{\mathrm{t}} = I$ and $diag(\lambda_1, \cdots, \lambda_i)$ is a square diagonal matrix with diagonal elements, $\lambda_1, \cdots, \lambda_i$. Therefore, to make C_i isotropy, it needs to be transformed to

$$\bar{C}_i = diag(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}) U^{\mathrm{T}} C_i U diag(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}) \ . \tag{4.16}$$

Therefore, a normalised point $\bar{p}$ is obtained from $\bar{p} = s U diag(\sqrt{\lambda_1}, \sqrt{\lambda_2}) \vec{p}$, where s is a

scaling factor and $\vec{p}$ is a point belonging to an elliptical IR. A bilinear interpolation is used to estimate the intensity value of non-integer $\bar{p}$ and followed by Gaussian blurring.

4.4.3 IR descriptor

The SIFT descriptor extracts distinctive feature vectors from the gradients of each normalised IR and the proposed method adopts the same SIFT implementation as in [77, 85]. The SIFT descriptor consists of two processes: reorientation and local histogram estimation. In the reorientation process, all gradient directions are reoriented in terms of the dominant orientations, which are estimated from an orientation histogram of a normalised IR patch. This reorientation makes the descriptor invariant to rotation. Note that an IR may have multiple dominant orientations, i.e., dominant orientation means all orientations having more than 80% of the votes in the orientation histogram. When estimating the histogram, the orientation of a gradient is weighted by its magnitude and value of a spatial Gaussian function centred at a normalised IR centre.

In the local histogram estimation, a $N_p \times N_p$ normalised IR is divided into 16 local image tiles and a 8-bin orientation histogram is estimated in each tile. Thus, a 128-by-N_d histogram matrix is obtained in each normalised IR (where N_d denotes the number of dominant orientations). The SIFT detector is normally applied to a textured IR but it may also be used with a MSER, and is particularly referred to as a shape descriptor in [85]. A shape descriptor has been shown to be better than the general SIFT descriptor in matching scene with near occlusion [85]. Some example of the MSER detector and its normalised patches are illustrated in Figure 4.6.

4.4.4 Clique descriptor

Although the shape descriptor and SIFT descriptor perform well in the general case, the matching performance can be further enhanced if nearby descriptors are also used. Thus, Lowe proposed a pair descriptor that groups two shape descriptors by the k-th nearest neighbour MSER [85]. The proposed clique descriptor extends this IR grouping concept by using all the neighbours simultaneously for matching instead of pairwise matching. Furthermore, the neighbour distance is appropriately weighted according to

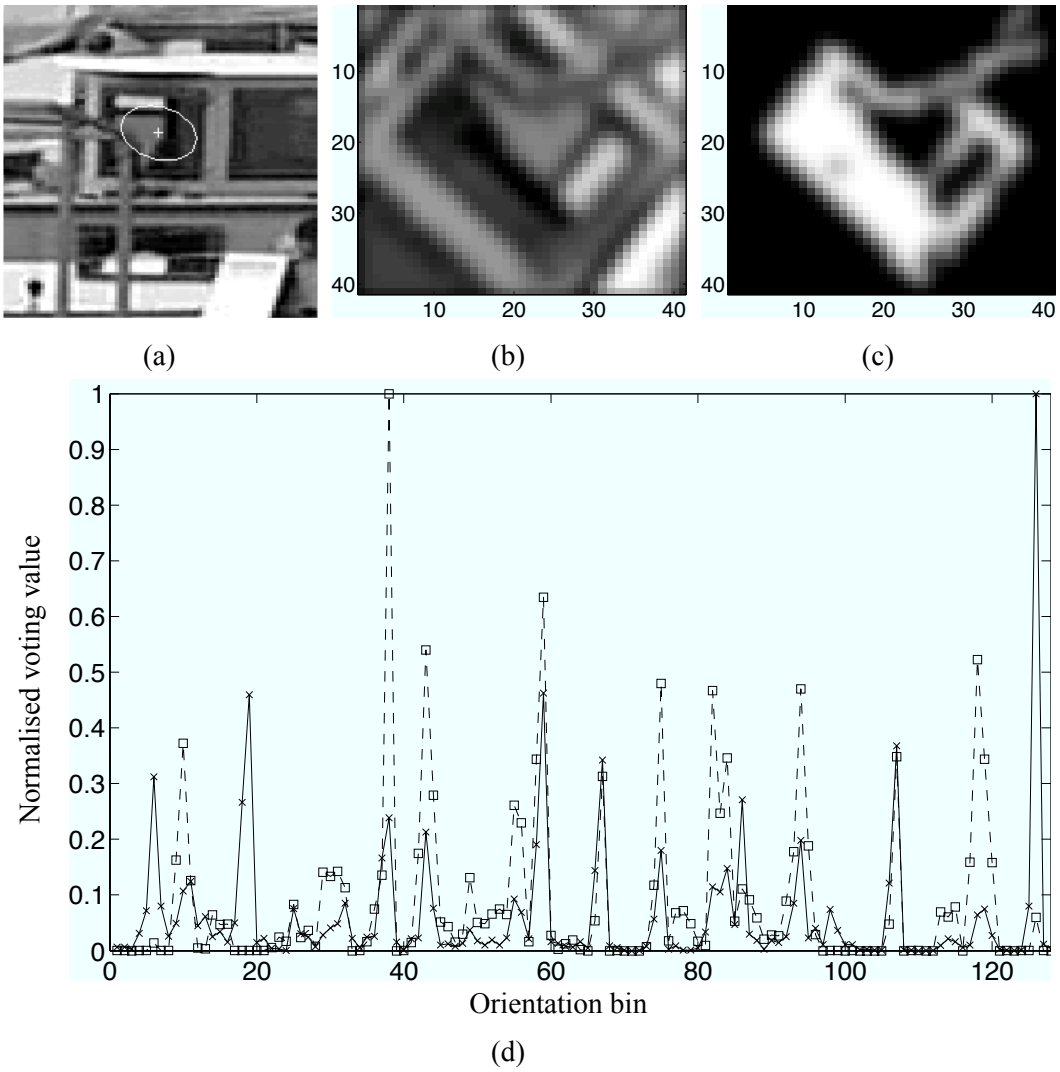


Figure 4.6: Example of MSER detection: (a) A detected MSER is illustrated as an ellipse and a cross represents the centre of the ellipse; (b) Textured MSER of (a); (c) MSER of (a); (b) and (c) are represented in a normalised patch of which $N_p = 41$; (d) The SIFT description of (b) and (c) are represented by a dashed line with a square mark and a solid line with a cross, respectively.

local geometry and the size of the ellipse.

To determine the local neighbours of a MSER $\mathcal{E}_i$, all MSER's need to be transformed to a local reference frame derived from $\mathcal{E}_i$. This is because we assume that the configuration of locally adjacent IR's is not changed significantly by an affine transform, and the entire feature distribution also contributes to form a local neighbour. All centres of MSER are thus transformed to a new space defined by the centre of $\mathcal{E}_i$ and its two eigen vectors of C_i. Since this causes the selection of neighbours to be too sensitive to small variation in centre position if $\mathcal{E}_i$ has high anisotropy ratio, MSER's with significantly small ellipses or high anisotropy ratio are excluded from the clique descriptor estimation. The transformed points are then tessellated by Delaunay triangulation. The i-th clique is uniquely defined by a local point cluster centred at $\bar{m}_i$, a normalised mean of $\mathcal{E}_i$.

For example, suppose that a set of transformed MSER centres in the local reference frame of $\mathcal{E}_i$, is denoted as $\mathcal{V}_i = \{\bar{m}_k \mid \bar{m}_k = T\vec{m}_k + \vec{m}_i,\ k = 1, \cdots N_\mathrm{t}\}$, where $T = sUdiag(\sqrt{\lambda}_1, \sqrt{\lambda}_2)$, $\vec{m}_k$ is the k-th mean of a MSER $\mathcal{E}_k$ and N_t is the total number of MSER in an image. The k-th clique in the i-th local frame $\mathcal{C}_i(k)$ then consists of a seed point $\bar{m}_k$ and the adjacent points directly connected to the seed. Thus, a clique centred at a point $\bar{m}_k$ in the i-th local frame is given by

$$\mathcal{C}_i(k) = \{\bar{m}_k\} \cup \{\bar{m}_j \mid \forall(\bar{m}_k, \bar{m}_j) \in \text{Delaunay edges}\} \ . \tag{4.17}$$

The proposed clique descriptor is designed to store all SIFT descriptors of MSER's in the same clique. Moreover, the angles defined by every two neighbours and a seed in the local frame, and the normalised size of neighbour ellipses are also stored for weighting the influence of neighbours. Therefore, a clique descriptor of a MSER $\mathcal{E}_i$ has three sets: a descriptor set, angle set and size set. A descriptor set $\mathcal{D}_i(k)$ of $\mathcal{C}_i(k)$ is defined by

$$\mathcal{D}_i(k) = \{F_j \mid \bar{m}_j \in \mathcal{C}_i(k)\} \ , \tag{4.18}$$

where F_j is a $128 \times N_d$ SIFT descriptor matrix of a MSER $\mathcal{E}_j$ whose mean is $\vec{m}_j$, and its

angle set is given by

$$\mathcal{A}_i(k) = \left\{ \theta_j \mid \theta_j = \cos^{-1}\left(\frac{(\bar{m}_{n1} - \bar{m}_s) \cdot (\bar{m}_{n2} - \bar{m}_s)}{|\bar{m}_{n1} - \bar{m}_s||\bar{m}_{n2} - \bar{m}_s|} \right) \right\} \ , \tag{4.19}$$

where $\bar{m}_s$ is a seed of $\mathcal{C}_i(k)$ and $\forall(\bar{m}_s, \bar{m}_{n1}, \bar{m}_{n2}) \in \mathcal{F}_i$. Finally, a size set is defined by

$$\mathcal{Z}_i(k) = \{ z_j \mid z_j = \frac{d_{\mathrm{s}}(j)}{d_{\mathrm{s}}(k)}, \forall \bar{m}_j \in \mathcal{C}_i(k) \text{ and } j \neq k \} \ , \tag{4.20}$$

where $d_{\mathrm{s}}(j) = \lambda_{j1}\lambda_{j2}$ and λ_{j1} and λ_{j2} are two eigen values of C_j. In short, (4.18) is a set notation of SIFT descriptor matrices in a clique and (4.19) represents a set of angles estimated from a clique. (4.20) represents a set of normalised sizes of ellipses in a clique. Figure 4.7(a) illustrates a Delaunay graph obtained in the local reference frame of the 254-th MSER having 7 neighbours in its clique. The textured MSER and MSER patches in the neighbourhood are shown in Figure 4.7(b), where T.M. stands for a textured MSER. An angle set describing a convex shape of a clique and the normalised size of neighbour ellipses are used for weighting factors. They are shown in Figure 4.7(c) and (d).

4.4.5 Clique descriptor distance

A new distance is required to match two corresponding cliques with different number of neighbours and it should be robust to false neighbours in a clique. Hausdorff Distance (HD) satisfies these two criteria, i.e., it defines a distance between two point sets without point correspondences and is robust against noise or outliers [54]. The general HD is a directional distance and the clique HD is

$$d_{\mathrm{hd}}(\mathcal{C}_i(m), \mathcal{C}_j(n)) = \max_{\bar{m}_\alpha \in \mathcal{C}_i} \min_{\bar{m}_\beta \in \mathcal{C}_j} \{ d_{\chi^2}(F_\alpha, F_\beta) \} \ , \tag{4.21}$$

where $d_{\chi^2}(\cdot)$ is a χ^2 distance that returns the minimal distance between two SIFT descriptor matrices, i.e.,

$$d_{\chi^2}(F_\alpha, F_\beta) = \min_{j,k} \frac{1}{2} \sum_{i}^{128} \frac{|F_\alpha(i,j) - F_\beta(i,k)|}{F_\alpha(i,j) + F_\beta(i,k)} \ . \tag{4.22}$$

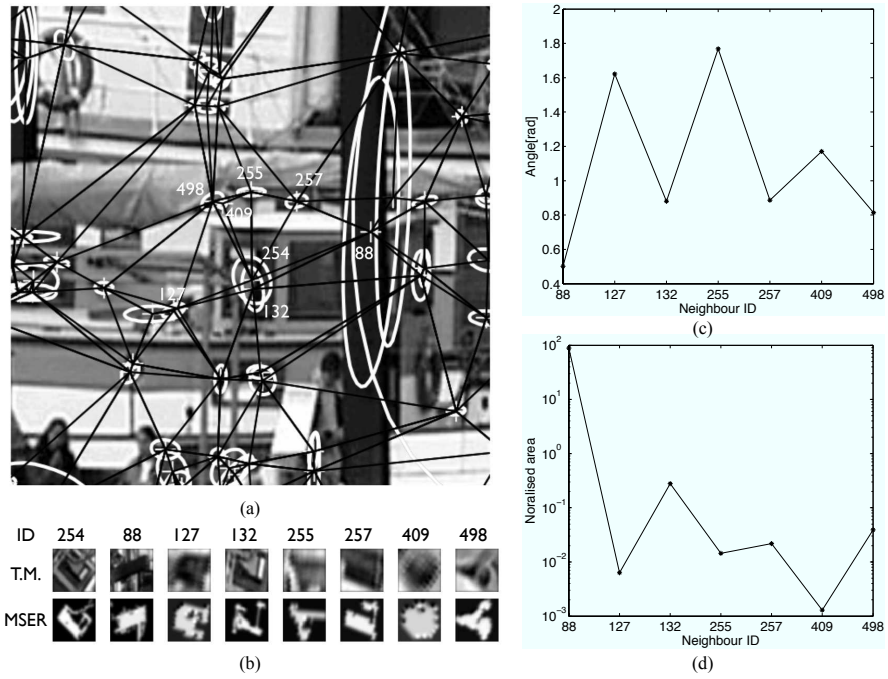


Figure 4.7: A clique descriptor: (a) a Delaunay graph determined by local reference frame of the 254-th MSER; (b) 7 neighbours of seed 254 in a clique, where T.M. represents a textured MSER; (c) and (d) show angle values in $\mathcal{A}_{254}(254)$ and size values $\mathcal{Z}_{254}(254)$

Thus, a non-directional HD is obtained by combining two directional distances. There are various ways to combine, e.g., averaging, weighted averaging, minimum and maximum of two directional distances. However, the maximum of two directional distances is the best for identification [70]. Thus, we define a non-directional HD, $d'_{\mathrm{hd}}(\mathcal{C}_i(m), \mathcal{C}_j(n)) = \max\{d_{\mathrm{hd}}(\mathcal{C}_i(m), \mathcal{C}_j(n)), d_{\mathrm{hd}}(\mathcal{C}_j(n), \mathcal{C}_i(m))\}$.

If a pair descriptor is obtained by simply appending the k-th nearest neighbour to a seed descriptor and (4.22) is used as metric, this distance is equivalent to the minimum distance between a seed distance and a neighbour distance, i.e., $\min(d_{\chi^2}(F_{s1}, F_{s2}), d_{\chi^2}(F_{n1}, F_{n2}))$. However, in this case the discriminating power is low because a seed distance is sometimes replaced with its closer neighbour distance. Furthermore, even though the sum of two distances is used it may be less distinctive than a single seed distance in some cases. For example, a pair-shape descriptor matching do not always perform better than a normal SIFT or a shape matching [85]. Therefore, the neighbour distance is appropriately weighted for the best performance, i.e., (4.21) which treats a seed and

neighbour distances equally needs to be a weighted distance,

$$d_{\mathrm{w}}(\mathcal{C}_i(m), \mathcal{C}_j(n)) = d_{\chi^2}(F_m, F_n) + w_{\mathrm{t}} d'_{\mathrm{hd}}(N_i(m), N_j(n)) \ , \tag{4.23}$$

where $N_i(m) = C_i(m) - \{\bar{m}_m\}$ and a neighbour weight

$$w_{\mathrm{t}} = w_{\mathrm{m}}\left(\frac{d_{\mathrm{hd}}(A_i(m), A_j(n))}{a_{\max}} + \frac{d_{\mathrm{hd}}(S_i(m), S_j(n))}{s_{\max}}\right)/2 \ , \tag{4.24}$$

where w_{m} is a maximum neighbour weight which is normally set to 0.5, and $a_{\max}$ and $s_{\max}$ are the maximum area and size distance between two images, respectively. Thus, if (4.24) only includes w_{m}, all neighbour distances are equally weighted by w_{m}. This equally weighted neighbour distance gives better matching than the adaptively weighted neighbour distance of (4.24) when the clique neighbours of a corresponding pair are not changed significantly.

In the proposed clique matching Modified HD (MHD) d_{mhd} replaces the general HD in (4.21) to improve matching performance and a directional clique MHD is given by

$$d_{\mathrm{mhd}}(\mathcal{C}_i(m), \mathcal{C}_j(n)) = \frac{1}{|\mathcal{C}_i|} \sum_{\bar{m}_\alpha \in \mathcal{C}_i} \min_{\bar{m}_\beta \in \mathcal{C}_j} \{d_{\chi^2}(F_\alpha, F_\beta)\} \ . \tag{4.25}$$

As a result of clique matching, Tentative Correspondences (TC's) are formed by collecting every matching pair of which the ratio of the best and second best clique distance is smaller than the threshold, and RANSAC optimises a TC with epipolar constraints because it is possible for TC's to have false correspondences.

4.5 Experimental results

This section presents experimental results of two proposed matching methods: graph matching and clique descriptor matching. In the first subsection, various tests are conducted to demonstrate the performance of the proposed graph matching, such as

- comparison of partial matching performance of the proposed graph matching with other traditional HD's;

- demonstrating identification capability of the proposed and generic HD matching. In this test a distance matching measure has been used.

On the other hand, the last subsection presents the test results of the proposed clique descriptor matching methods. These tests include comparing the number of inliers obtained from EWC, AWC, SIFT, pair descriptor, correlation matching when

- query image is affected by unknown 3D camera motion;
- query image has homogeneous texture in addition to unknown 3D camera motion;
- query image is transformed by zoom and rotation;
- query image is affected by an approximated circular motion.

These two sets of tests reveals that the proposed methods performs better than the conventional matching methods.

4.5.1 Similarity invariant graph matching

To evaluate the performance of the proposed graph-based method in partial matching, an experiment uses a 348×360 grey level image of a cup, from which the Harris detector extracts 251 model points. Figure 4.8(a) shows the Delaunay graph of the model points overlaid on the image. To create a portion of the model data, a reference line and a sweeping line are used in the test. The reference line is parallel to the horizontal axis and has an intersection with the mean point of the model data. The sweeping line sweeps the model points in a clockwise direction at a sweeping angle θ_s measured from the reference line. The sweeping angle determines the size of the selected portion.

Figure 4.8(b) shows a selected test data with $\theta_s = 60°$, and its Delaunay graph. Since the general HD is able to cope with partial matching, the performance of HD and its variants in partial matching are compared with that of the clique HD for $\theta_s = 20°$ to $\theta_s = 340°$. To quantify partial matching performance, distances of each matching method is computed with respect to the sweeping angle. Model data was obtained and displayed in Figure 4.8(a), while the test data is selected as a portion of the model data [see Figure 4.8(b)]. As mentioned, the size of the portion is controlled by a sweeping angle (i.e., test data is closer to the model data as the sweeping angle increases). Because the generic HD

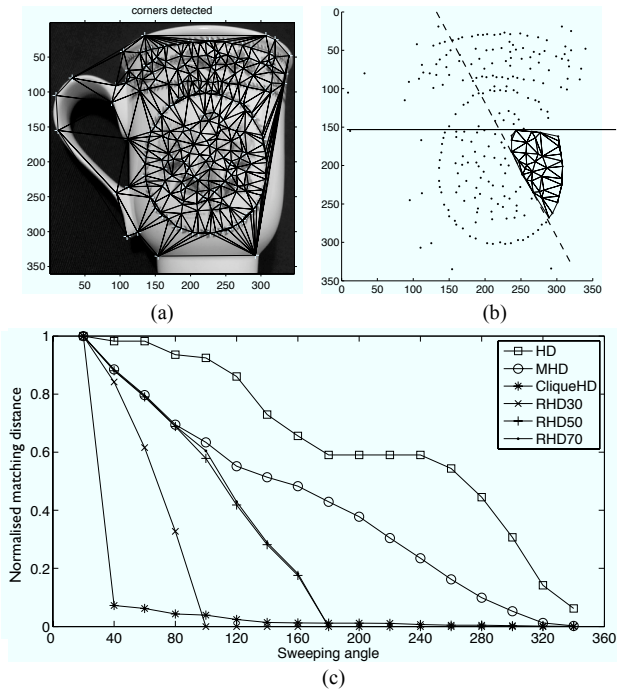


Figure 4.8: (a) The 251 model points extracted from the 348×360 image for partial matching are denoted by +, with their Delaunay graph overlaid. (b) The selected data is bounded by the solid reference line and the dashed sweeping line with $\theta_s = 60°$. (c) Normalised matching distances of HD, MHD, 30% RHD, 50% RHD, 70% RHD and clique HD, are respectively denoted by □, ○, x, +, • and *.

matching gives pixel distance between two point clusters (i.e., model and test data), while the proposed graph matching distance is defined as the difference of cross ratios which are dimensionless, direct comparison is not possible. To overcome this, all distances obtained from various sweeping angles are normalised by the maximum distance and the results are then compared [see Figure 4.8(c)].

Despite the small portion of model data selected with $\theta_s = 60°$, the clique HD matching distance of 0.589 is the best, whilst HD, MHD, 30% Ranked HD (RHD), 50% RHD and 70% RHD scores are respectively 0.953, 0.786, 0.388, 0.782 and 0.783. This is because the local graphical information is not changed. RHD's score 0 as the size of the portion is close to the model data because they only use the best ranked score. However, this characteristic is not desirable for identification.

Images of three small objects as shown in Figure 4.9 are used for evaluating the identification capability of the proposed method. Two of the objects have a similar shape

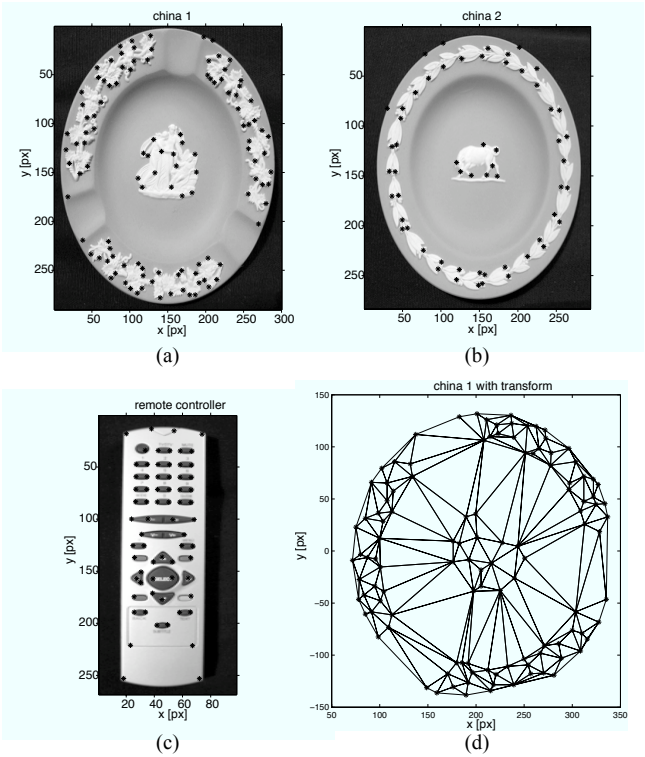


Figure 4.9: Test images for evaluating the identification capability: (a) china 1; (b) china 2; (c) remote controller; and (d) a graph generated by rotating (a) by 45° and random translation.

except for details of their decoration [see Figure 4.9(a) and (b)] and the last test image is generated from Figure 4.9(a) using a similarity transform. The images in Figure 4.9(a), (b) and (c) respectively generate 113, 57 and 68 feature points.

The normalised matching distances are summarised in Table 4.1 where a, b, c and d respectively correspond to the objects in Figure 4.9(a), (b), (c) and (d). Since the overall shapes of object (a) and (b) are similar, the HD score (i.e., distance) for these two objects is relatively small. On the other hand, the clique HD gives a larger distance for object (a) and (b) because their local details are different, even though the general shape is not. When a model and test data are related by a similarity transform, clique HD scores 9.83 whereas general HD scores 60.7 (see the matching distance between a and d in Table 4.1), i.e, the HD matching score between (a) and (d) is significantly increased but clique HD gives a small distance under the transform. These results show that the clique HD achieves a much better performance in identifying an object, e.g., for model based

Table 4.1: Identification test results.

	clique HD				HD			
	a	b	c	d	a	b	c	d
a	0	49.41	38.61	9.83	0	11.92	81.88	60.07
b	49.41	0	100	44.12	11.92	0	73.76	59.89
c	38.61	100	0	40.58	81.88	73.76	0	100
d	9.83	44.12	40.58	0	60.07	59.89	100	0

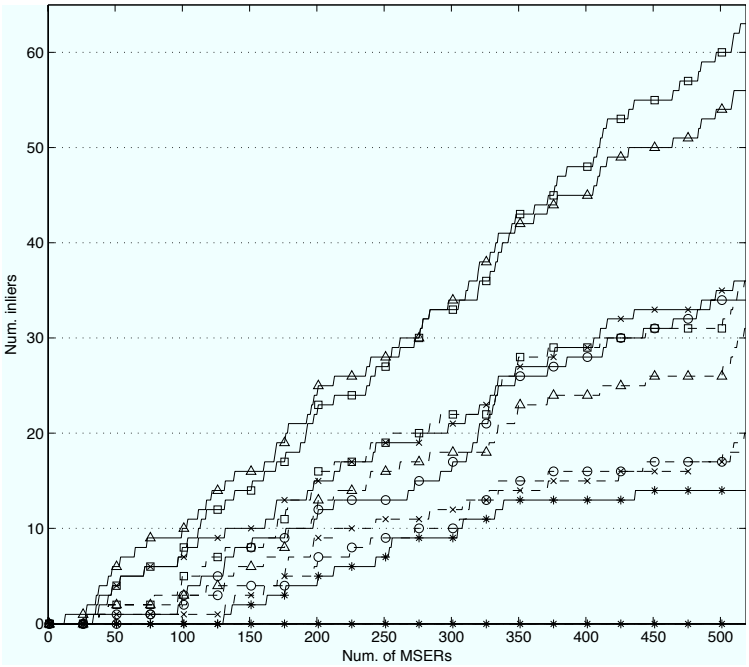
matching. However, when the general shape of two objects are considerably different, e.g., the rectangular remote controller and the circular china, traditional HD performs better and the average distances of clique HD and HD in this case are 59.73 and 85.21, respectably (see column c in the table).

4.5.2 Clique descriptor matching

In the valuation of matching performance of the clique descriptor, a pair descriptor uses a SIFT description of the third nearest neighbour in the local reference frame of a seed IR and the TC threshold is set to 1.4. This means a tentative pair of the i-th IR in an image is determined when the ratio of the best and the second best distance in the i-th row vector of a distance matrix is greater than 1.4. Since TC does not permit multiple correspondences, only the closest distance is selected as a tentative correspondence. As a measure of matching performance, the number of inliers from TC's is counted when using group descriptors: a pair, Equally Weighed Clique (EWC) and Adaptively Weighted Clique (AWC) descriptor matching. The larger the number of inliers the better is the matching. The results of a single descriptor matching (SIFT) and a correlation-based matching are also included respectively as the reference performance and the result without descriptor.

The first test compares the results of matching images captured at different views. In two 640×480 images of a tea shop [see Figure 4.10(a)], 519 and 546 MSER's are detected. The corresponding points in the two images are connected by lines. In general, matching based on textured MSER's generates more TC's than matching with MSER's because texture in an IR gives significant clues for matching unless an image has homogeneous texture. 61 inliers are detected in the textured MSER matching whilst 37 inliers are found in the MSER matching. Both best results are obtained when the EWC descriptor is used [see Figure 4.10(b)]. In this case, 71 and 44 TC's are estimated before RANSAC,

(a)

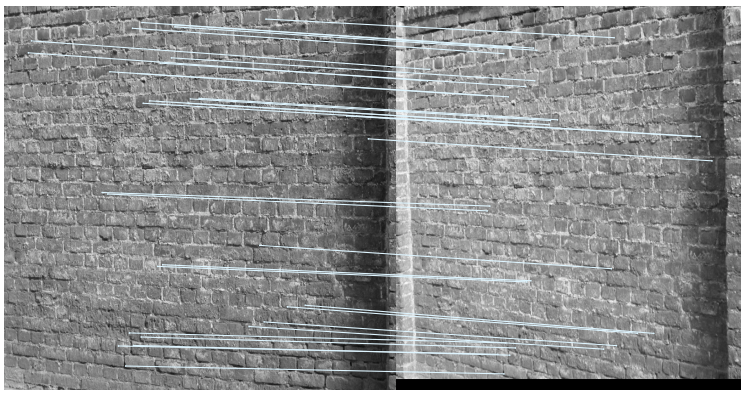


(b)

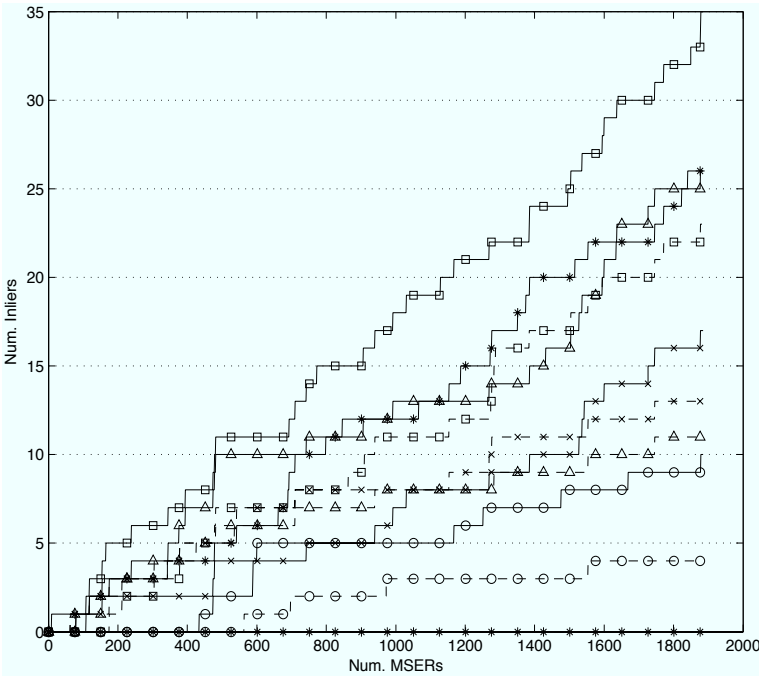
Figure 4.10: Matching images from different views: (a) Result of equally weighted clique descriptor matching using textured MSER's; (b) Inliers from matching using: EWC descriptor (□), AWC descriptor (Δ), a pair descriptor (∘), SIFT (x) and correlation (*). A solid and dashed lines denote matching result of textured MSER's and MSER's, respectively.

i.e., 86% and 84% of TC's are classified as inliers. A pair descriptor performs less well than SIFT in both textured MSER and MSER matchings, even though a pair descriptor matching of textured MSER's has the same number of TC's as the SIFT matching. The inlier ratios of a pair and SIFT descriptor matchings are 71% and 73%, respectively. Without a descriptor, correlation-based matching detects 14 inliers when textured MSER's are used for matching. However, 1 inlier has been found without texture. The performance of AWC descriptor matching lies between the SIFT descriptor matching and EWC descriptor matching. Both weighted group descriptor matchings perform better than general SIFT descriptor matching. This result shows that an additional neighbour distance increases the discrimination power of a single descriptor if neighbour distance is appropriately weighted when the configuration of neighbourhood is not significantly changed.

The second test compares matching performances when an image has homogeneous texture. The wall images [see Figure 4.11(a)] from the Oxford data set [87] are used. Matching using SIFT descriptor and correlation give similar performance, i.e., SIFT description of a textured MSER is not more distinctive than a textured MSER without descriptor. However, the proposed distance improves the performance of SIFT descriptor matching. 1885 and 1656 MSER's are detected due to the larger size of the test images (1000×700). However, the performance is more degraded than that of the first test, e.g., the total inliers of the best matching methods are reduced to less than half of the best result in the first test. The SIFT matching of textured MSER's detects 17 inliers from 18 TC's while correlation matching detects 26 inliers out of 31 TC's, i.e., 94% and 84% inlier ratios are obtained. However, without texture information, correlation matching cannot detect any correspondence. On the other hand, since the neighbourhood does not change significantly, two weighted neighbour distances result in the most TC's; 35 and 26 correspondences are detected out of 42 and 29 TC's by EWC and AWC, respectively. A pair distance simply adds a descriptor at the k-th nearest position from a seed IR. Thus, if the additional descriptor is not distinctive, the addition of two descriptors does not improve the matching performance. However, the EWC descriptor increases the chance of being distinctive by adding more than one neighbour description. Furthermore, the AWC descriptor penalises neighbour distance according to the shape of two matching



(a)



(b)

Figure 4.11: Matching images with homogeneous texture from different views: (a) Result of equally weighted clique descriptor matching using textured MSER's; (b) Inliers from matching using 3 group descriptors, SIFT and correlation denoted using the same symbols in Figure 4.10(b).

cliques.

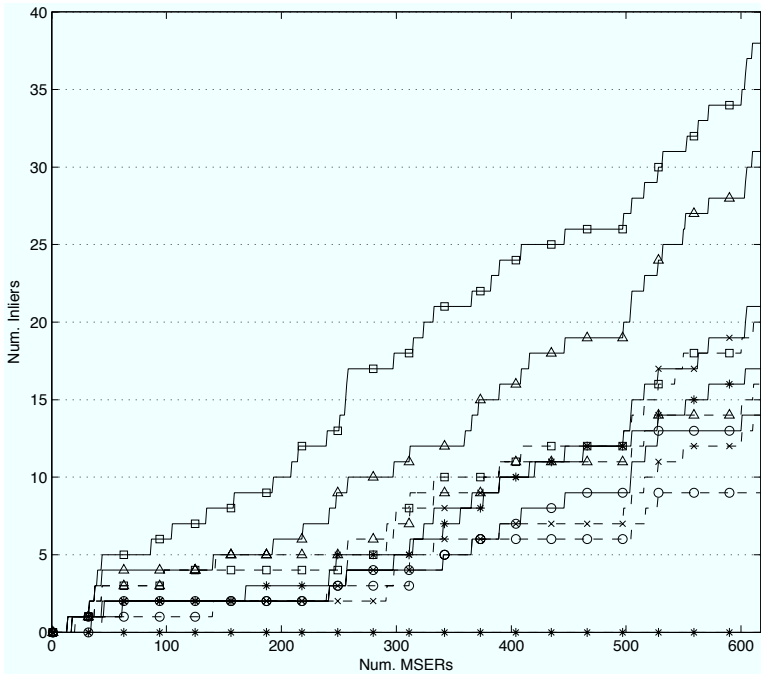
The shape of a clique, e.g., $\mathcal{A}_i(k)$ and $\mathcal{S}_i(k)$ are not altered by camera zoom and 2D rotation because a Delaunay graph is invariant under a similarity transform, i.e., scaling, rotation and translation. Thus, the third test evaluates any effects of these camera operations on matching. An image and its zoom-out and rotated version [see Figure 4.12(a)] from the Oxford data set are used. After removing small and high anisotropic IR's, each image produces 617 and 653 MSER's. However, since the MSER detector is not scale invariant, without multi-scale MSER detection as in [85] the matching result is degraded. The EWC descriptor matching detects 38 inliers from 54 TC's, AWC descriptor matching detects 31 inliers from 43 TC's, and SIFT matching detects 21 inliers from 26 TC's.

The proposed matching method is required to detect as many inliers as possible from images generated from a circular motion in order to produce a large number of widely distributed 3D-2D pairs, which increases the stability of projection matrix estimation. The fourth test evaluates the matching of images from circular motion using eight images captured at every 6° rotation from 0° to 40° [see Figure 4.13(a)]. Only matching results of textured MSER's are compared, since matching with texture information is generally better than without it. Due to the use of a black background, relatively small number of MSER's (around 130) are detected in each image. The projections of a planar object with a small rotation and without rotation are similar, i.e., the influence of affine distortion is small. Thus, the performance of all matching methods do not vary with rotation. EWC detects 56 inliers with 80% inlier ratio while SIFT matching detects 45 inliers from 56 TC's. As affine distortion increases, the performance of all methods also decreases. In particular, AWC descriptor matching detects more inliers than EWC descriptor matching after 30°, which approaches the SIFT result after 36° rotation. This is because the neighbour configuration changes significantly as the rotation increases. However, AWC is still better than SIFT.

4.6 Conclusions

HD-based point matching has advantageous characteristics, e.g., it is capable of matching without exact point pairs and partial matching, and it is robust against noise and outliers.

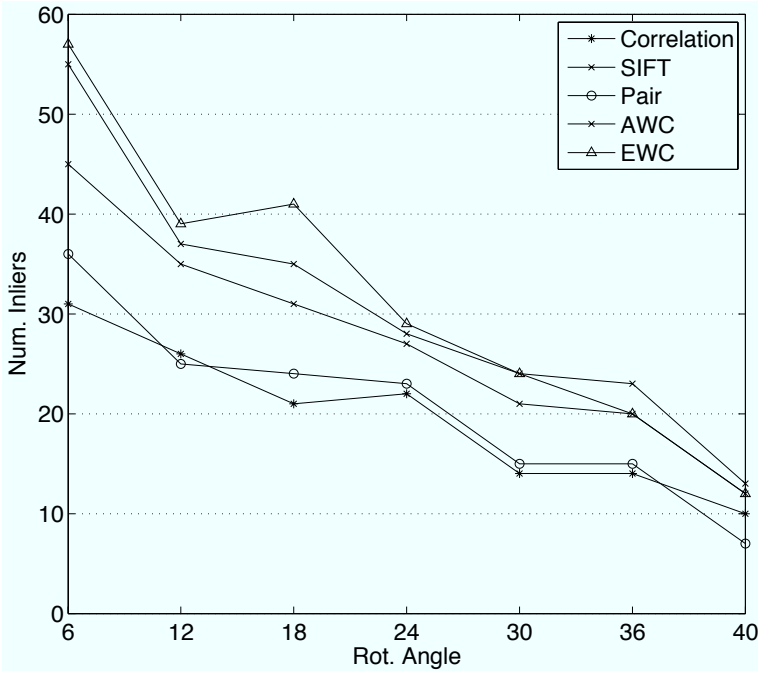
(a)



(b)

Figure 4.12: Matching two images one of which with zoom and rotation: (a) Result of the equally weighted clique descriptor matching using textured MSER's; (b) Inliers from matching using 3 group descriptors, SIFT and correlation denoted using the same symbols in Figure 4.10(b).

(a)



(b)

Figure 4.13: Examples of matching images generated from a circular motion: (a) images at 0° to 40°; (b) textured MSER based matching results using correlation, SIFT, Pair, AWC and EWC.

Therefore, many variants of HD have been introduced. This chapter proposes a clique HD that incorporates a geometrical distance in the traditional HD matching framework. The proposed method performs matching at local point sets called cliques, which are uniquely formed by the Delaunay tessellation.

To achieve similarity invariance, the first method uses a set of cross ratios that is defined by four collinear points on every boundary edge in a clique. The normalised area of a face in a clique is used for weight when matching two triangles. Experiments show that the proposed similarity invariant graph matching method is robust to noise and outliers, which normally would deteriorate the performance of graph-based matching methods. However, the method cannot account for apparent shape matching (e.g., matching by the profile of an object) because it is solely based on features from local entities. After all, the invariant properties only hold up to a similarity transformation.

The second distance explores a method that improves matching performance by grouping local feature descriptors called a clique descriptor which can achieve more TC's with a high inlier ratio than the previous SIFT matching. An individual descriptor is distinctively defined in the affine invariant regions used in traditional SIFT descriptor. Since the MSER detector is adopted for a region detector in the proposed method, two types of normalised regions, i.e., textured MSER's and MSER's are evaluated for matching evaluation, and we found that the proposed grouped SIFT descriptor on textured MSERs generally performs best amongst other matching methods.

Chapter 5

Visual hull refinement

5.1 Introduction

In the case either that the silhouette images only capture a certain part of an object or that the number of silhouette images is insufficient to provide adequate 2D constraints for an approximation of the object shape, the resulting VH is larger than the actual object volume. Therefore, a large number of object images captured at various camera positions are generally required in order to approximate a VH closer to the actual shape of the object. However, when the number of images involved in an initial object reconstruction is restricted but additional images are available after the initial reconstruction, the initial VH can be improved further from these additional images. In other words, if an additional object image is provided as well as its projection transform, the image (even when captured at different time) can improve the VH.

One condition imposed on an additional image is that its projection transform should be defined in the same 3D world frame of the initial VH. In addition, there is no assumption on the internal parameters of the projection matrix like a circular motion, i.e., a new projection should account for updated internal camera parameters such as focal length, pixel skewness, and the centre of an image plane. Therefore, a straightforward method to retrieve a new projection matrix is to calibrate an additional image by means of 3D-to-2D point correspondences defined in the initial VH framework. Although a self calibration technique [88, 89, 90] can also determine the camera parameters, in this par-

ticular case, it is more beneficial to exploit known projection transforms and geometrical constraints between a new image and the initial images used for the initial VH.

A related research proposed in [91, 92] improves a VH by aligning several coarse VH's created at different time instances. Thus, it is necessary for this approach to estimate a 3D rigid motion between VH's. Assuming that camera internal parameters are known (or unchanged) across time, the 3D motion is estimated from colour consistent points[1] on each coarse VH. However, this colour consistency is only reliable when many views are involved in the coarse reconstruction, i.e., this approach is not appropriate for improving a coarse VH from one (or few) additional image. To address this problem, this chapter investigates an alternative approach where 2D features are shared among multiple views for the VH improvement, because sufficient information has already been embedded in the initial images of a coarse reconstruction. Consequently, it proposes an algorithm that calibrates an additional image from 2D point correspondences.

The proposed algorithm calibrates an additional image from an image triplet, in which two images are involved in an initial reconstruction (i.e., their projection matrices are known). Thus, the linear triangulation [8] can produce true 3D points from point correspondence in these two images. 2D points associated with the true 3D point in an additional image are located by the intersection of two epipolar lines. Thus, this method needs to define point correspondences between every two images in an image triplet, but they do not need to be viewed by all images[2]. The clique descriptor matching presented in Chapter 4 is incorporated in the proposed method to enhance the matching performance when an additional image is distorted by an unknown 3D motion. Furthermore, it is more robust to colour variation due to a feature descriptor used in the clique descriptor matching.

This chapter is organised as follows. Section 5.2 explains the epipolar transfer, which is a main idea for searching 2D points in an additional image for the calibration. Section 5.3 presents the overall algorithm which collects calibrating data from image triplet. Finally, the experimental results and conclusions are respectively presented in Section 5.4 and 5.5. Thus, this chapter is mainly dedicated to explain following research

[1]The colour consistent points are called a coloured surface point in [91].

[2]If point features are viewed by all three images, a trifocal tensor can be defined, instead of three fundamental matrices.

aspects:

- brief review of the epipolar transfer in three view geometry;
- coarse visual hull refinement algorithm from additional image;
- the performance of calibration result from the proposed method.

5.2 Epipolar transfer

Figure 5.1 illustrates a three-view geometry. Given three views [see I_1, $I_{(i+1)}$, and $I_{(i-1)}$], in which a corresponding pair $\vec{x}^{(i-1)} \leftrightarrow \vec{x}^{(i+1)}$ is established in two views $I_{(i+1)}$ and $I_{(i-1)}$, the epipolar transfer can locate the matching point $\vec{x}^{i}$ in the third view by the intersection of two epipolar lines (see two dotted lines on I_i). As explained in Appendix B, these epipolar lines are determined by a fundamental matrix of a pair of stereo images, i.e., a fundamental matrix is a transform that relates a point from one image to an epipolar line in the other image [3]. Suppose that a fundamental matrix F_{12} is a 3×3 singular matrix that maps a point in $I_{(i-1)}$ to an epipolar line in I_i, and similarly F_{32} is defined from two views $I_{(i-1)}$ and I_i, i.e., $(\vec{x}^{(i+1)})^{\mathrm{T}} F_{23} \vec{x}^{(i)} = 0$. Then, the point $\vec{x}^{i}$ which satisfies $\vec{x}^{i} \leftrightarrow \vec{x}^{(i+1)}$ and $\vec{x}^{(i-1)} \leftrightarrow \vec{x}^{i}$ is obtained by

$$\vec{x}^{i} = [F_{12}\vec{x}^{(i-1)}]_{\times} F_{32} \vec{x}^{(i+1)}. \tag{5.1}$$

Therefore, locating a matching point using (5.1) in an image triplet is called an epipolar transfer.

Although the epipolar transfer is a straightforward method to implement (i.e., only two fundamental matrices are sufficient to transfer a point), it will fail to locate an exact point when two epipolar lines are parallel in the additional image. This particular case occurs when a camera centre of a new image lies on the same epipolar plane $\vec{\pi}_e^{w}$, defined by $\vec{o}_1^{w}$, $\vec{o}_3^{w}$, and $\vec{x}^{w}$. To avoid this degeneracy, two fundamental matrices should be extended to a trifocal tensor, which is a more stronger three-view constraint defining a point-to-line correspondence in an image triplet [3]. However, the estimation of a trifocal tensor requires 2D point correspondences of three images and the use of tensor notation

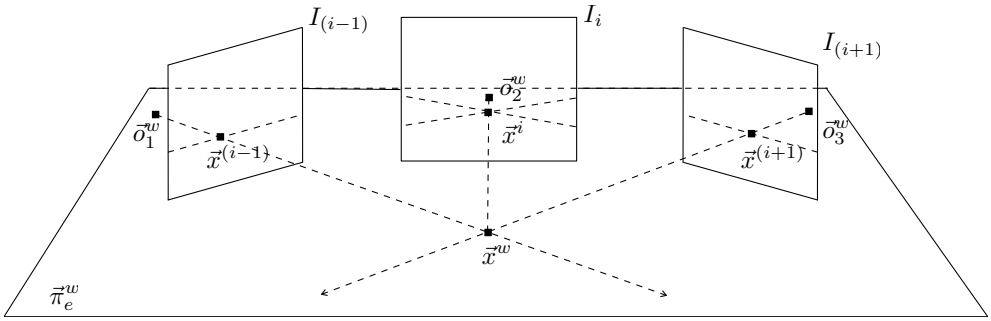


Figure 5.1: Geometry of an image triplet: given a point correspondence $\vec{x}^{(i-1)} \leftrightarrow \vec{x}^{(i+1)}$, the linear triangulation can infer its 3D position $\vec{x}^w$ by intersecting two rays back-projected from the corresponding points (see dotted arrow on an epipolar plane $\vec{\pi}_e^w$). Furthermore, when two fundamental matrices F_{12} and F_{32} are known, an image of $\vec{x}^w$ in an additional image I_i can be determined without a projection matrix by the intersection of two epipolar lines.

is more complicated than that of a fundamental matrix. Since this thesis presumes that initial images are obtained from a circular motion and additional images are captured at different time, it is highly unlikely for the centre of a new camera accidentally fall into the same epipolar plane defined by the initial images.

Figure 5.2 shows an example of a degenerate case of the epipolar transfer. Three images shown in Figure 5.2(a), (b), and (d) are captured at 6°, 0°, and 12° in a circular motion, and (a) and (c) are images captured at the same rotational angle. Fundamental matrices are estimated from each pair of images shown in each column of Figure 5.2, and the epipolar lines associated with a point marked + in (b) and (d) are illustrated as white lines in (a) and (c). Since the camera centres of all three images reside in the same epipolar plane, the two epipolar lines are almost identical[3]. Therefore, when a corresponding pair is given in two images (b) and (d), the matching point in (a) cannot be located. To avoid this degenerate case, one of the camera centres in an image triplet should not be on the same epipolar plane as other two, and this condition is more easily met than the condition of a pure rotation in practice.

[3]Ideally, two epipolar lines should be identical but errors of feature detection and a fundamental matrix estimation result in little difference.

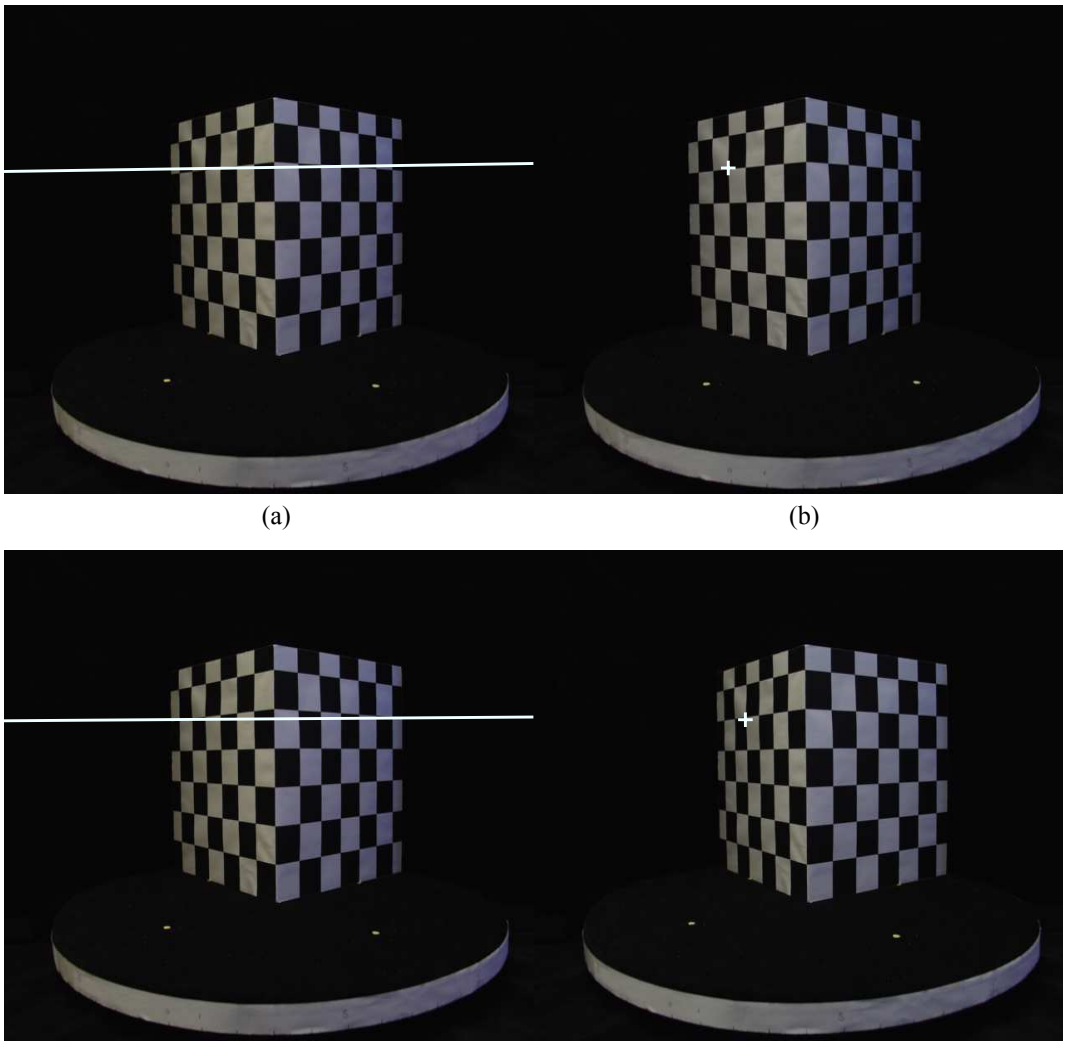


Figure 5.2: Example of a degenerate epipolar transfer: three images (a), (b), and (d) are captured in a circular motion, and (a) and (c) are images captured at the same rotational angle. Two epipolar lines associated with a point marked as '+' in (b) and (d) are represented as a white line in (a) and (c). Those two lines are parallel, so that an intersection of two epipolar lines cannot locate a matching point.

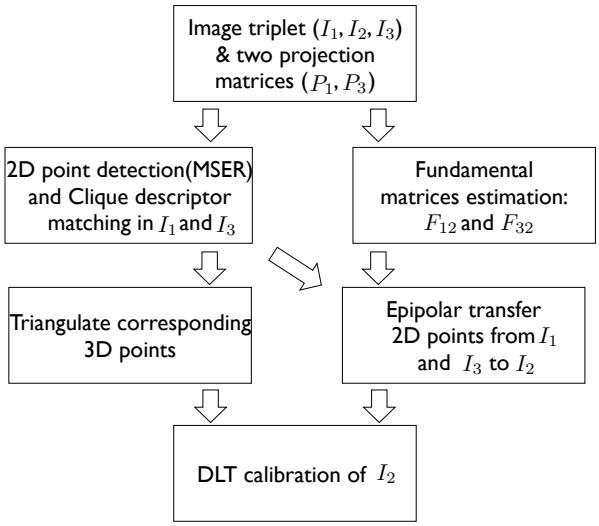


Figure 5.3: An algorithm for the image calibration from an image triplet.

5.3 Image calibration algorithm

An algorithm proposed in this section calibrates a new image from an image triplet, in which two images have been used for an initial volumetric reconstruction, i.e., their two projection matrices are also given as an input to the algorithm. A fundamental DLT method (presented in Chapter 3) is utilised for the calibration in the proposed algorithm. Therefore, the input data is processed to reconstruct 3D points in the world frame of the initial reconstruction as well as to locate their projection in a new image. This 3D-to-2D point correspondences are achieved by linear triangulation and epipolar transfer.

Figure 5.3 shows the block diagram of the proposed method, which requires three images (I_1, I_2, and I_3) and two known projection matrices (P_1 and P_3) as previously mentioned. Before the estimation of corresponding data, the MSER detector locates affine invariant regions in each image and SFIT represents the detected regions as a distinctive feature vector. The clique descriptor matching presented in Chapter 4 then estimates the correspondences among three pair of images, i.e., (I_1, I_2), (I_2, I_3), and (I_1, I_3). The correspondences from the first two pairs of images are used to estimate the fundamental matrices F_{12} and F_{32} whilst the other correspondences from the last image pair are used for the stereo reconstruction, which subsequently constructs a set of 3D points for the additional image calibration.

The linear triangulation used in the proposed algorithm is the fundamental depth reconstruction method used in the stereo vision. For example, from a pair of point correspondences in two view $\vec{x}^{(i-1)} \leftrightarrow \vec{x}^{(i+1)}$ (see Figure 5.1), a 3D point $\vec{x}^w$ associated with the correspondences is determined by the intersection of two rays, which are illustrated as dotted arrows on $\vec{\pi}_e^w$. In other words, the projection of $\vec{x}^w$ onto $I_{(i-1)}$ should coincide with an image of the point, i.e.,

$$[\vec{x}^{(i-1)}]_\times P_1 \vec{x}^w = 0, \tag{5.2}$$

where P_1 denotes a projection matrix of $I_{(i-1)}$. Similarly, an image of $\vec{x}^w$ defines another condition. Therefore, a 3D point $\vec{x}^w$ is achieved from a system of linear equation that combines these two conditions, i.e.,

$$\begin{bmatrix} x\vec{p}_1^{3\mathrm{T}} - p_1^{1\mathrm{T}} \\ y\vec{p}_1^{3\mathrm{T}} - p_1^{2\mathrm{T}} \\ x'\vec{p}_3^{3\mathrm{T}} - p_3^{1\mathrm{T}} \\ y'\vec{p}_3^{3\mathrm{T}} - p_3^{1\mathrm{T}} \end{bmatrix} \vec{x}^w = 0, \tag{5.3}$$

where $\vec{x}^{(i-1)} = [x\ y\ 1]$, $\vec{x}^{(i+1)} = [x'\ y'\ 1]$, and $\vec{p}_m^n$ represents the n-th row vector of the m-th projection matrix P_m.

On the other hand, the projections of these reconstructed 3D points in I_2 are achieved from two fundamental matrices F_{12} and F_{32} and point correspondences defined in a pair of images I_1 and I_2. Once the two epipolar lines are constructed from each correspondence, the epipolar transfer completely defines the corresponding 2D points. Finally, the DLT calibration estimates a projection matrix of I_2 from the estimated observations.

5.4 Experimental results

This section presents the test results of the proposed VH improvement method. To show the performance of the algorithm, following two tests are conducted:

- to improve a coarse VH by adding additional uncalibrated image.This experiment exploits epipolar transfer results of an image triplet;

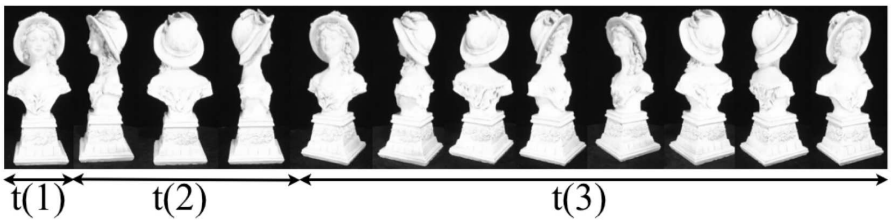
- more comprehensive calibration tests are performed. In this test, the estimation of the proposed method is compared with the reference values that is obtained from a 3D calibration rig.

Figure 5.4 illustrates some examples of a VH and its surface mesh with respect to the number of views used. Suppose that a different number of bust images are added across time as shown in Figure 5.4(b), i.e., 1, 3 and 8 images are added at $t(1)$, $t(2)$ and $t(3)$, respectively. Then, the quality of volumetric approximation is enhanced as time passes. The VH's corresponding to $t(n)$ are respectively shown from the first to the third columns of Figure 5.4(b), where the first and second row shows the voxel views and their surface view. The reference VH is generated using sixty bust images with the seven-iteration of an octree construction and is shown in the fourth column of Figure 5.4(b). The reference reconstruction is carved from a 30[cm]×30[cm]×30[cm] initial voxel, i.e., the minimum 3D resolution of a VH is 30×2^{-7}[cm]. The Marching Cubes[4] (MC) followed by 3D smoothing are applied to estimate the reference object surface mesh.

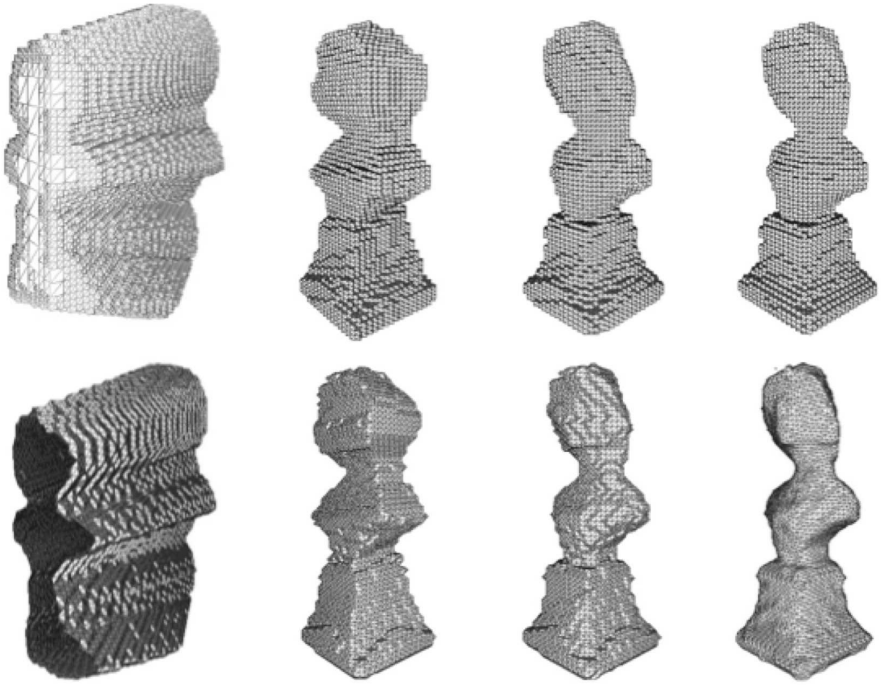
The quantitative results of the reconstructions are summarised in Table 5.1 which shows the volume (Vol.) in [cm^3], the number of voxels (Vox.) and triangle meshes (Face) of the VH's created using added images in $t(1)$ to $t(3)$ while the reference VH is shown in the last column indexed as 'Ref.'. The table shows that increasing the number of silhouettes improves the quality of a VH. For example, the first column of Figure 5.4(b) is the intersection of single silhouette cone with the initial cube. Thus, the shape of an object is not apparent due to the large amount of redundant volume. However, adding three images taken at four main diagonal positions of the scene configuration improves the shape reconstruction and the volume is reduced to 1682.14[cm^3] from 9101.17[cm^3]. Moreover, the volume is continuously decreased as an image is added, and the reconstruction using 12 images is almost similar to the reference obtained from 60 images. This result also demonstrates that reasonably similar SfS reconstruction can be achieved from a coarse reconstruction with a small number of additional image taken at critical position.

Other experimental results of the epipolar transfer are shown in Figure 5.5(a)-(c). Two initial images shown in Figure 5.5(a) and (c) are taken from a circular motion

[4]A conventional surface construction algorithm from a volume data. More details of the Marching Cubes are presented in Chapter 6.



(a)



(b)

Figure 5.4: (a) Images added to VH construction across time $t(n)$, where $n = 1, 2, 3$; (b) VH's (the first row) and their surface mesh (the second row) generated using 3 views and with added images. The reference VH and surface meshes are shown in the last column.

Table 5.1: VH's created using different number of additional images.

	$t(1)$	$t(2)$	$t(3)$	Ref.
Vol.	9101.17	1682.14	1432.89	1349.05
Vox.	28579	9626	8109	7813
Face	35288	12649	10676	10376

at rotation angle 246° and 264°, respectively. The adaptively weighted clique matching of textured MSER's detects 35 TC's from two images with 3D rotation, and RANSAC removes false correspondences from the TC's. These TC's are visualised as white crosses in Figure 5.5(a) and (c), and the corresponding 3D positions estimated by the triangulation are shown as dots in Figure 5.5(f).

The fundamental matrices from the image in Figure 5.5(a) to the image in Figure 5.5(b), and from the image in Figure 5.5(c) to the image in Figure 5.5(b) are estimated from image correspondences from each pair of images [see the matching results in Figure 5.5(d) and (e)]. The epipolar transfer of white crosses from the two initial images to the additional image is illustrated in Figure 5.5(b), where the two epipolar lines of a point are shown as white lines. Thus, the image in (b) is calibrated using the estimated 3D-to-2D pairs. The three cameras and the corresponding 3D points are visualised in Figure 5.5(f), where a square, asterisk, and circle respectively represent the camera position of the images in (a), (b) and (c), and cross and dots respectively show the 3D origin of the world coordinate and 3D points.

Once the camera matrix is determined, an object location in an additional image (b) is confined by the convex hull of the projection of the initial VH. An active contour [23] using the convex hull as initial position or a thresholding followed by morphological operation on the hull completes the search for the object. Four images captured from a circular motion at 162°, 246°, 264° and 354° produce an initial VH of a spray in Figure 5.5(h) and the projection of the initial VH onto the additional image is shown in Figure 5.5(g), where dots represents corner points of initial voxels and lines indicate a convex hull formed from them. The final carving result is shown in Figure 5.5(i), where the handle of the spray becomes more detailed and the body area is also reduced.

The last experiment compares the projection matrix estimated from a DLT method using a 3D calibration rig with the result of the proposed method, where 3D points are estimated from the two views used for the initial VH and the corresponding 2D points are found by epipolar transfer among three views. 2D projections of 91 3D corner points of a checkerboard pattern shown in Figure 5.6(a)-(f) are manually selected and used to estimate the reference projection matrices. The proposed method uses two known projection matrices of Figure 5.6(a)-(b), and fundamental matrices obtained from 2D-to-2D

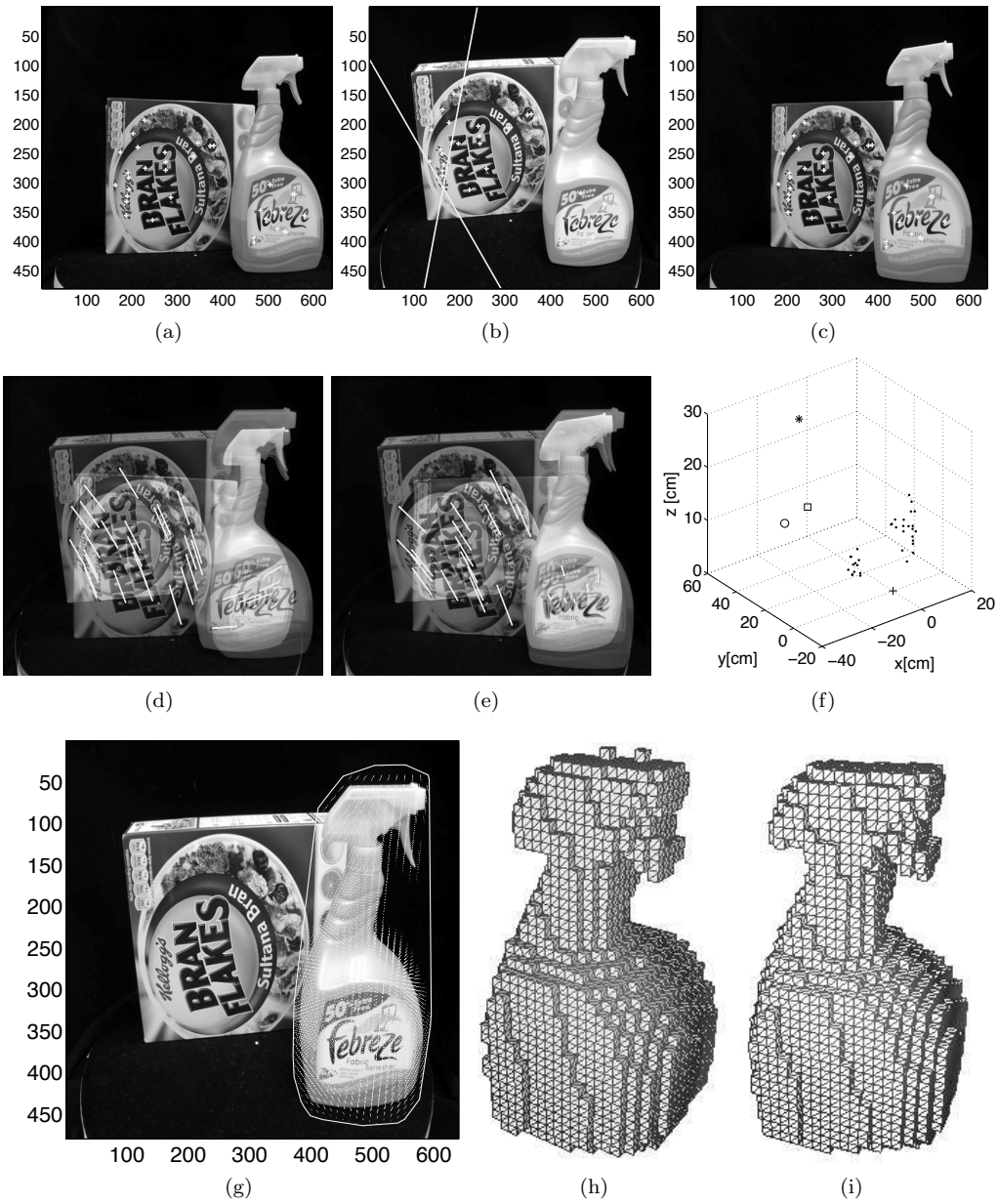


Figure 5.5: (a) and (c): Two initial images taken from a circular motion with + denoting matched MSER centres; (b) An additional image with the corresponding points in the initial images; (d) and (e) are point matching result for estimating two fundamental matrices; (f) 3D positions of corresponding points, and camera positions for images in (a), (b) and (c) are respectively marked as □, ∘ and *; (g) Projection of the initial VH of a spray onto the new image, where dots represents voxel corners and lines visualise a convex hull defined from dots; (h) Initial VH constructed from four images including the initial images; (i) Improved VH by adding the new image.

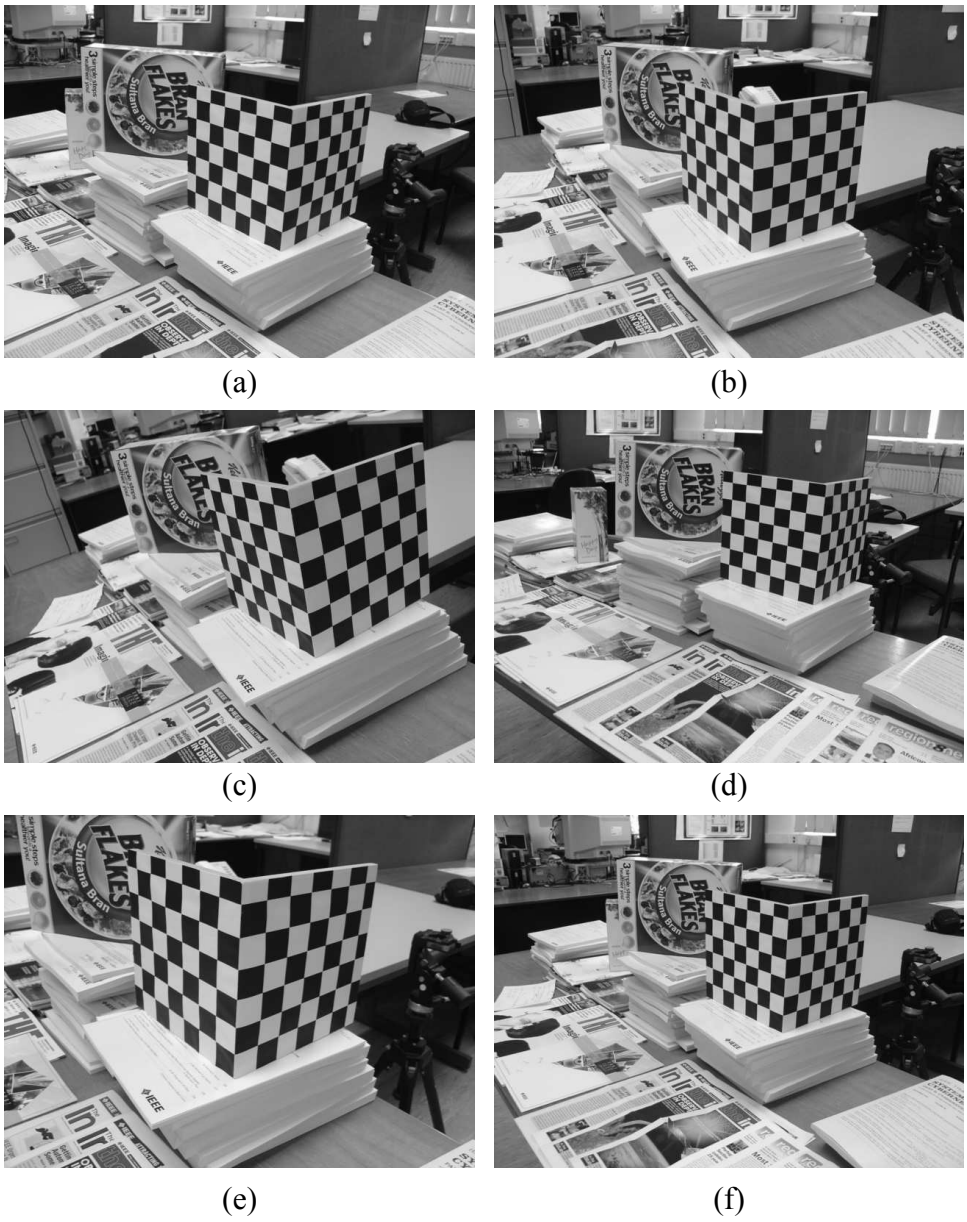


Figure 5.6: Calibration test: (a) and (b) initial images with known projection matrices; (c)-(f): images with unknown projection matrices. The intrinsic and extrinsic parameters of the camera for the images (c)-(f) are listed in Table 5.2

Table 5.2: Estimated camera parameters from DLT Vs. the proposed method

	(c)	(d)	(e)	(f)
f_x	641.146 / 598.478	669.045 / 605.174	625.636 / 595.767	593.095 / 589.847
f_y	621.938 / 597.541	713.143 / 606.554	627.978 / 592.016	591.197 / 588.781
s_k	-6.086 / 2.508	-7.8666 / 1.759	10.083 / 2.413	1.558 / 4.054
c_x	391.737 / 366.512	271.793 / 369.246	317.061 / 332.564	369.992 / 358.166
c_y	286.757 / 217.287	172.846 / 212.976	221.755 / 213.711	212.348 / 220.715
t_x	-24.710 / -22.987	-71.838 / -62.864	-24.990 / -23.595	-44.638 / -44.514
t_y	-36.726 / -34.991	-37.024 / -29.664	-29.279 / -27.074	-42.317 / -42.029
t_z	31.317 / 29.970	36.053 / 27.527	28.017 / 27.256	30.357 / 30.047

point correspondences which are found by the EWC descriptor matching method, when estimating an unknown projection matrix.

The results are shown in Table 5.2, where four intrinsic and three extrinsic parameters of the camera are extracted from each estimated projection matrix of Figure 5.6(c)-(f). The first and second values represent the results of the proposed and reference method, respectively. There is some difference between the two results especially for the intrinsic parameters. This is due to inaccurate 3D positions from the initial images and erroneous fundamental matrices, which result in erroneous 2D positions. However, these two are conflicting conditions to satisfy simultaneously. For example, the linear triangulation used in most stereo reconstruction methods becomes more correct when two rays from the initial images are orthogonal, i.e., requiring a wide baseline between two views. However, when a wide baseline is used, the number of corresponding pairs decreases which degrades the fundamental matrix estimation. Since the fundamental matrix estimation is based on 2D-to-2D correspondences, when matching points are not equally distributed over an image or the number of pairs are insufficient, the fundamental matrix is only correct in the region where point pairs are obtained, i.e., the epipolar transfer is sensitive to the location of 2D-to-2D correspondences.

5.5 Conclusions

This chapter explores a method that improves a coarse VH by using additional images. If the two fundamental matrices between an additional image and the two initial images are known, the projection matrix of an additional image can be determined. Therefore, a

silhouette cone produced from the additional image can confine the initial VH closer to the actual object. However, the estimation of a fundamental matrix requires 2D-to-2D point correspondences, which are not robustly detectable when there is a significant 3D camera motion. To address this, the clique descriptor matching method (presented in Chapter 4) is used in the proposed image calibration algorithm. However, the epipolar transfer is not always stable due to the degenerate configuration of three views. Moreover, a fundamental matrix is only reliable when the sufficient number of 2D-to-2D correspondences are widely distributed in an image.

Chapter 6

Robust surface modelling

6.1 Introduction

The Marching Cubes (MC) are the most successful method for surface construction from an octree [12]. It estimates surface triangles from intersection octants, and the location of the triangles are determined by the configuration of inside vertices of an intersection octant. However, the MC generated surface may contain unexpected holes or discontinuities that are not present in its octree. One reason for surface discontinuity is due to the connectivity of octants as was first reported by Mercier et. al. [24] who also proposed a process which thickens the intersection octants to ensure 6-connectivity, and changes an inside octant to an intersection octant. But the process is not straightforward to implement for the following reasons: since it checks whether two adjacent inside and background octants are in the same hierarchy level of the octree, the octree hierarchy is repeatedly referred to when creating a surface; and the subsequent image pixel based refining method has to verify whether two adjacent surface lines remain connected.

Another possible reason for discontinuity is due to the topological ambiguity of the MC algorithm. For example, when a face of an octant has an intersection point with a surface in each of its four edges, the topologically correct connections among the intersection points become ambiguous and these result in Type A hole problem [93, 94], and Chen et al. reported seven ambiguous configurations that create holes and incorrect connectivity [95].

A more practical reason for surface holes is due to erroneous camera calibration and imperfect silhouettes, which change the position of the projection of an octant or the value at the projection position. Therefore, traditional octree construction methods take special care of these processes. Szeliski used adaptive thresholding followed by a local shrinking operation for silhouette detection, and a hexagonal calibration pattern is attached to the turntable for a precise camera calibration at every rotation [21]. Mercier and Meneveaux used over-exposed images and a seed-fill algorithm to generate silhouette images and attach an LED on the rotational axis of the turntable for accurate calibration [24]. However, measurement error is inevitable in calibration and there are no image preprocessing algorithms that can deal with all effects of imaging conditions. For example, a seed-fill algorithm can reduce noise on silhouette images but at the expense of losing concave surface details.

On the other hand, octree construction is robust against image noise because an octant is not removed when any nonzero-valued pixel is found within the projection of the octant. Thus, despite some unpredictable error on silhouette images, the resulting octree can be similar to the octree created from error-free silhouette images if the size of an octant is not too small. The octree construction only changes the status of an octant from inside to intersection. Therefore, to retain concavities in a convex VH, simple thresholding[1] is preferable for its octree construction. However, in this case, its MC surface is significantly degraded.

Thus, this chapter proposes a surface construction method for an imperfect MC result. The method exploits the connectivity information of an octree, which is referred when building a new face from imperfect MC vertices. It is premised that a general non-convex object can be represented as a piecewise convex set, and an object surface is constructed from an aggregate of its local convex surfaces. The initial MC vertices are grouped into different slices and classified, and connections are made with appropriate vertices in adjacent slices in order to determine local convex regions. The Bayes rule is used for classifying and connecting the MC vertices. The conditional probability density functions (pdfs) used by the Bayes rule are estimated from octree vertices that are

[1]If *a priori* knowledge regarding the shape of an object is provided in advance, it is possible to manually modify silhouettes or uses the size filtering algorithm presented in Chapter 2.

regarded as sampled points on the true 3D object.

A similar method which uses data slices for surface generation has also been proposed in [96]. However, the cylindrical mapping of this method focuses on merging 3D range data obtained from different views and only a simple object is considered. The principal axis of such an object must pass through the object, and a normal of the principal axis must pass through only one point on the object, i.e., the object is convex. An alternative mapping procedure is also proposed for a more complex object, e.g., an object with a single cavity like a cup. Nevertheless, the algorithm has not been designed for a general object. Thus, if there are multiple clusters in a slice then the algorithm will have difficulty in aligning the slices.

This chapter is organised as follows. Section 6.2 presents some existing surface construction algorithms, including MC and its variants, 3D convex hull method and the Delaunay triangulation. The possible problems with these methods are also explained. An overview of the proposed method is detailed in Section 6.3. Section 6.4 presents the proposed local hull-based surface construction method. Finally, Section 6.5 and 6.6 present the experimental results and conclusions. In summary, this chapter focusses on the following research topics:

- literature review of current surface construction algorithms and their problems;
- the proposed robust surface construction, which is based on the local convex hull algorithm;
- experimental results of the proposed method and comparison with the traditional approaches.

6.2 Surface from silhouettes

6.2.1 Marching cubes and its variants

MC [12] was originally developed for 3D visualisation of medical images, e.g., computed tomography or magnetic resonance images but through its simplicity has evolved to other applications [13, 97]. MC cannot predict the implicit surface directly from intersect octants, assuming that intersection octants may include an actual surface which crosses

an edge joining two vertices of a surface octant with opposite status, i.e., inside and background. Thus, when MC constructs surface patches, it connects the middle of the edges having different status. The novelty of the MC is that it reduces the possible 256 configurations of inside vertices to 14 cases by assuming reverse and rotation symmetries. However, the original algorithm is concerned with how to define a surface when the inside status is clearly classified. Thus, when the decision on an inside vertex is affected by the precision of the projection matrix and the noise in a silhouette image, it may result in artefacts on surfaces.

In practice, a silhouette is generated by thresholding an image before the intersection test. Any error in the resulting silhouette is insignificant as far as constructing an octree because the 2D intersection test determines the status of an octant only if a convex polyhedron estimated from the projection of an octant intersects with an silhouettes, i.e., an octant is likely to be classified as the intersection even though all projections of eight points are not found in S_i (more details of the progressive intersection test are presented in Chapter 2). Thus, even if some points of an octant are erroneously classified as background, an octant cannot be carved out but only changes its status from inside to intersection in the worst case. However, the resulting error in the MC surface construction is significant since the locations of inside vertices are crucial in defining the iso-surface of an octant. For example, when the projection of an octant results in one vertex within a silhouette, i.e., case a as shown in Figure 6.1(a), the octant is classified as intersection and the inside vertex is easily identified. However, it is ambiguous to identify an inside vertex when only part of the edge of an octant is within a silhouette, i.e., case b as shown in Figure 6.1(b), or the silhouette is entirely within an octant, i.e., case c as shown in Figure 6.1(c), although the projection of the octant is classified as intersection. Case b and c are frequently found when imperfect silhouette images and projection matrices are used or when the octree resolution is not small enough.

If any projection of a 3D vertex is erroneously classified as background, it supersedes other statuses previously defined in other silhouettes [see (3.16)]. This erroneous classification often occurs if there is noise in the silhouette and no inside vertices are found even though the octant is classified as intersection, e.g., as in case b and c. Thus, the MC surface loses surface patches that result in holes and unattached object segments.

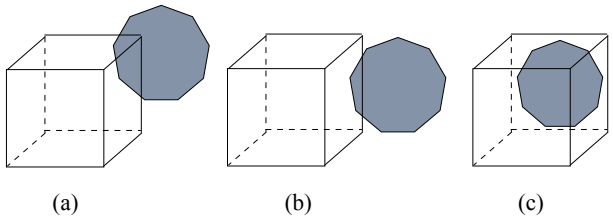


Figure 6.1: Example of three intersection cases : (a) case a; (b) case b; (c) case c. A polygon with grey shade represents a silhouette and a cube is formed from eight projection points.

To avoid this situation, the Voting MC (VMC) counts the number of cases classified as background and identifies a background vertex if the number of case (i.e., vote) is greater than a threshold τ_{v} [98]. Thus, the decision function (3.16) is revised as

$$\left\{\sum_{i=0}^{m} cost_{\mathrm{r}}(\theta_i, \vec{x}^w)\right\} - \tau_{\mathrm{v}} \leq 0, \quad 0 \leq \tau_{\mathrm{v}} < m, \tag{6.1}$$

where m represents the total number of images used and θ_i denotes the i-th rotation angle in a circular motion. For the VMC implementation, each vertex stores how many times it is classified as an inside vertex throughout all images in the fourth element[2] of the colour variable of each vertex. It does not need to account for all octants which fail to produce offspring in earlier generation, so that the voting process is only performed on octants with intersection status in the last generation of the octree. The code in Figure 6.2 presents the pseudo code of the voting process, where two iterations, i.e., per view (line 4) and per octant (line 12), are used. The function DecideInsideVertex in line 15 project a corner point onto the image obtained from line 8 and return the result which indicate whether it is inside or not.

The problem with VMC is that its result varies with the threshold level even for a convex object, and it is difficult to choose an appropriate threshold. For example, the simple convex object shown in Figure 2.10(a) loses surface triangles as the voting level decreases because low voting threshold can change the status of a background vertex to an inside vertex, which can change the status of an octant from the intersection to the inside. Figure 6.3 illustrates this effect. When the voting threshold is 100%, i.e., it is equivalent

[2]Each colour is stored in a double array with four elements in BGRA order (see COctant declaration in Chapter 2), in which the last alpha value is replaced with the vote counted in the VMC implementation.

```
//1. initialise count buffer of each vertex of an octant
//2. store the number of total generations in an octree in nGen
//3. count inside case
for(int i = 0 ; i < nFiles; i++)
{
    // image loading
    IplImage* imgBin = NULL;
    bRes = LoadImg(i,&imgBin);
    if (bRes)
    {
        // project and classify inside vertices belonging to
          // an intersection octants in the last generation
        for (int j = 0 ; j < m_Octree.GetPopulationInGen(nGen-1); j++)
        {
            if (m_Octree.GetOctant(nGen-1,j)->GetStatus() == INTERSECTION);
            DecideInsideVertex(imgBin, ppmatCamera[i], m_Octree.GetOctant(nGen-1,j));
        }
        // release image buffer
        cvReleaseImage(&imgBin);
    }
}
```

Figure 6.2: Voting estimation.

Table 6.1: VMC surface construction result

	m	$0.9m$	$0.8m$	$0.7m$
Vertices	10584	8701	2761	510
Faces	21156	10651	2177	412
Lost Oct.[a]	275	5674	9586	10620

[a]15552 intersection octants are obtained from the last generation of the seven-level of octree

to the traditional MC, it can produce closed surface patches as shown in Figure 6.3(a), i.e., due to the simplicity of the shape it is safe to use MC. However, when $\tau_{\mathrm{v}} = 0.9m$, it looses many patches, which creates holes and unattached surface triangles [see Figure 6.3(b)], and most of the patches disappear when $\tau_{\mathrm{v}} = 0.7m$ as shown in Figure 6.3(d). Table 6.1. summarises the the number of vertices, faces and lost octants. The number of lost octants indicates the number of intersection octants belonging to case b and c in the last generation that fail to produce surfaces.

6.2.2 Delaunay triangulation and convex hull

An alternative approach, the 3D Delaunay Triangulation[3] [56], constructs a surface by defining tetrahedrons from arbitrarily distributed 3D points, e.g., Figure 6.4(b) illustrates 3D Delaunay result from 100 randomly generated 3D points. The 3D DT characterises each tetrahedron by not allowing any point within its circumsphere. If there is such a point then DT subdivides the tetrahedron without changing the shape of a super tetrahedron

[3]This is an extension of the 2D Delaunay triangulation shown in Chapter 4 with a tetrahedron simplex

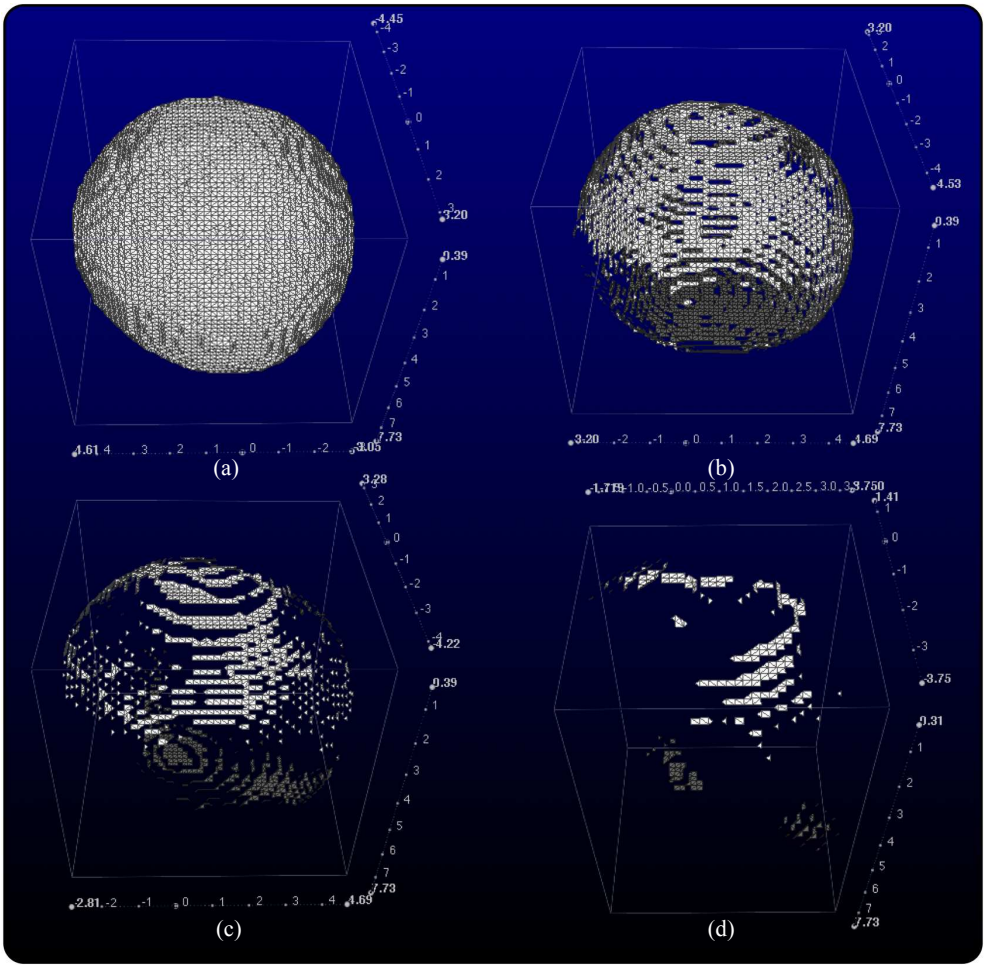


Figure 6.3: Example of VMC results relative to τ_v: (a) voting threshold is set to $\tau_v = m$, which is equivalent to mc result; (b)$\tau_v = 0.9m$; (c)$\tau_v = 0.8m$; (d)$\tau_v = 0.7m$. Three axes represents three bases of world fame and the unit is [cm].

[55]. The Constrained Delaunay Triangulation (CDT) [99] has been evolved in order to include a predescribed boundary. In 3D, however, CDT cannot tetrahedralise some special ployhedron without an additional point or surface modification, e.g., a twisted prism, and the problem in determining whether a given polyhedron can be tetrahedralised is NP-complete [100].

Although other variations of the DT algorithms, e.g., conforming constrained DT [101] and the conforming DT [102], have been proposed to solve the problem, they assume that initial boundary information is given. Furthermore, the result of DT in 3D is not triangles but tetrahedrons, i.e., three additional faces are redundantly created in order to make one surface triangle [compare the number triangles shown in Figure 6.4(a) with that shown in (b)]. Also, a protocol for the input vertices is required to avoid the degeneracy of DT, e.g., the rejection of the coincidency of two points and the coplanarity of four vertices, or the use of a slack variable to prevent rounding error.

DT is topologically related to a convex hull. If for a set of points I in the n-dimensional space, a set of the points I' are fitted to a hyper quadric in $n+1$ dimension, e.g., $x^2 + y^2 + z^2 = d^2$ for $n = 2$, then the projection of the convex hull of I' onto the lower dimension is equivalent to the DT result of I [64]. Algorithmically, the convex hull algorithm is simpler than DT and results in a fewer number of triangular patches because it only stores surface triangles, i.e., there are no internal triangles as normally found with DT. However, both algorithms are designed to construct convex shapes. Therefore, we propose a general surface construction algorithm which copes with concavity whilst preserving the advantages of the 3D hull algorithm. This is achieved by classifying local convexities from an imperfect MC surface and estimating each local hull using the 3D hull algorithm. Finally, locally constructed surfaces are combined to complete the surface construction. Thus, the problem of surface construction becomes two separate problems, i.e., how to classify a local convexity and how to construct a local surface.

6.3 Overview of the proposed method

The overall surface construction process is illustrated in Figure 6.5 where the proposed method involves the processes enclosed in two grey-shaded processing boxes, i.e., the local

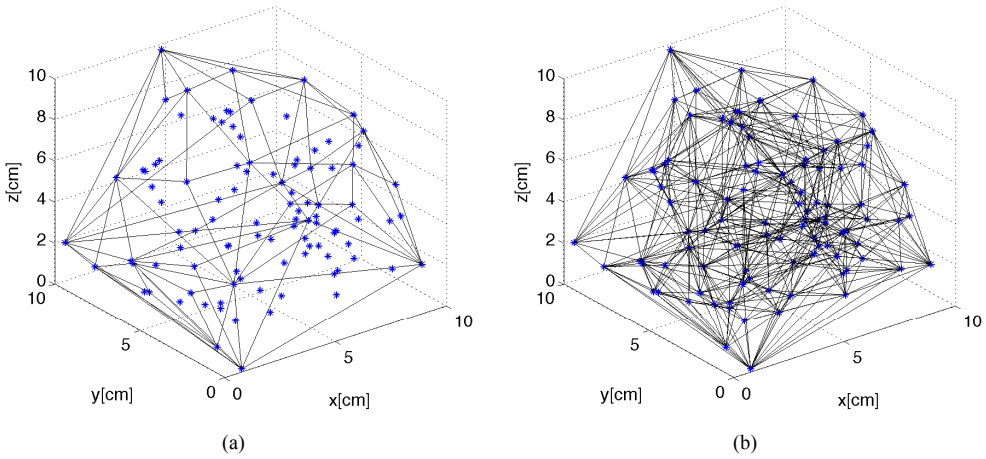


Figure 6.4: Convex hull Vs 3D Delaunay from randomly generated 100 3D points: (a) convex hull does not use all points to construct triangle patches but the result is always convex; (b) all points are connected by triangle edges and result also forms a convex.

convex classification and 3D hull based surface generation. The projection matrix estimation determines the projection matrix at the reference position and uses it to estimate other projection matrices in the circular motion, with intervals at 6° rotational angle. Images of an object are thresholded to generate silhouette images, and sixty projection matrices, one for each of the sixty image planes are fed to the initial data preparation process.

An octree data is first constructed in the initial data preparation process and it is used to estimate the initial MC vertices that are normally obtained from the best VMC. In the octree construction, a 2D intersection test is used, i.e., an octant is projected onto every silhouette image, and if it intersects a silhouette the octant is classified as intersection and split into eight sub-octants. For robust octree construction, the projection of an octant is approximated as a rectangle, and if the rectangle has no intersections then the corresponding octant is classified as background and is removed. There are numerous algorithms that facilitate the intersection test in 2D image planes. Szeliski proposed an almost real-time algorithm which uses a half distance transform [21]. Potmesil approximated the projection as a rectangle [20]. Chien and Aggarwal used a quad tree [22]. Ahuja and Veenstra reduced the number of silhouette images by only using orthographic views [32]. A few 3D intersection test algorithms have also been proposed in [29, 30]. However, since the speed of creating the volumetric data is not an issue in this research,

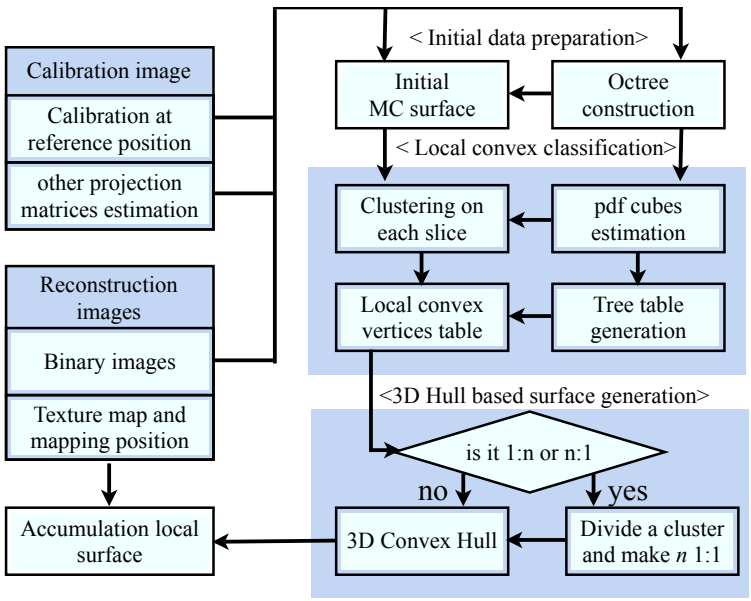


Figure 6.5: The overall surface construction process.

the fundamental octree construction method has been developed for the robust surface construction process.

The proposed local-hull based surface estimation comprises two sequential processing blocks: local convex classification, and 3D hull based surface construction. The first block determines local convex regions from the initial MC surface vertices. Data slicing and classification are required to define these convex regions. For the classification of initial vertices, cluster conditional pdf's are estimated from every octree slice. As a result of the first block, a tree table storing information on the cluster connection and local convex vertices table are passed to the next block.

The 3D hull-based surface construction block creates appropriate local surfaces using the convex hull algorithm. Local convexity with multiple connections, e.g., 1:n or n:1 connections are divided into n 1:1 local convexities before it is used to create a local hull. As a result of the second block, all local surfaces are aggregated to complete the surface estimation.

6.4 Local hull-based surface construction

Two properties of a 3D object are premised. The first is connectivity which assumes the surface of an object should cover an object tightly without any unattached object segment. If a surface is obtained without violating the first property, each edge of the surface should be traversed twice to make two connected patches. Otherwise there is a hole in the surface and the edge is called a dangling edge.

The second property due to the assumption of piecewise convexity of a 3D object is the continuity of an object. It allows a shape with local convexity to be similar to its adjacent convexity if they are connected [103]. To make this property more robust, an object needs to be sliced infinitesimally. However, each slice cannot be smaller than the size of the smallest octant. The second property enables the data distribution between two octree slices to be approximated. It considers a local convexity to be continuously connected to other local convexities in adjacent slices.

6.4.1 Volumetric data slicing

The proposed algorithm uses the best VMC vertices since they are closer to the actual surface than vertices of intersection octants, and the number of vertices are considerably reduced. On the other hand, the octree vertices are used to define a local convexity from MC vertices and their connections. In order to represent an object as a piecewise convex set, the data is sliced along the z axis and the slicing interval is defined by the height of the smallest octant, i.e., w. Even though a MC surface has twice finer resolution, the same slicing level is used to keep the correspondence between an octree and a MC slice. The sliced results are stored as a set of binary images planes called MC slices $\mathcal{S}^{\mathrm{mc}} = \{I_0^{\mathrm{mc}}, \cdots, I_i^{\mathrm{mc}}\}$, where i denotes z index. Similarly, the corresponding octree vertices are also grouped into slices and the results are stored in a set of octree slices, i.e., $\mathcal{S}^{\mathrm{oc}} = \{I_0^{\mathrm{oc}}, \cdots, I_i^{\mathrm{oc}}\}$.

Suppose that a function $obj(\cdot)$ indicates whether a VH contains a 3D point $\vec{p}^{w} = [x\ y\ z\ 1]^{\mathrm{T}}$, i.e.,

$$obj(\vec{p}^{w}) = \begin{cases} 1 & \text{if } \vec{p}^{w} \in \mathrm{VH} \\ 0 & \text{otherwise} \end{cases}. \tag{6.2}$$

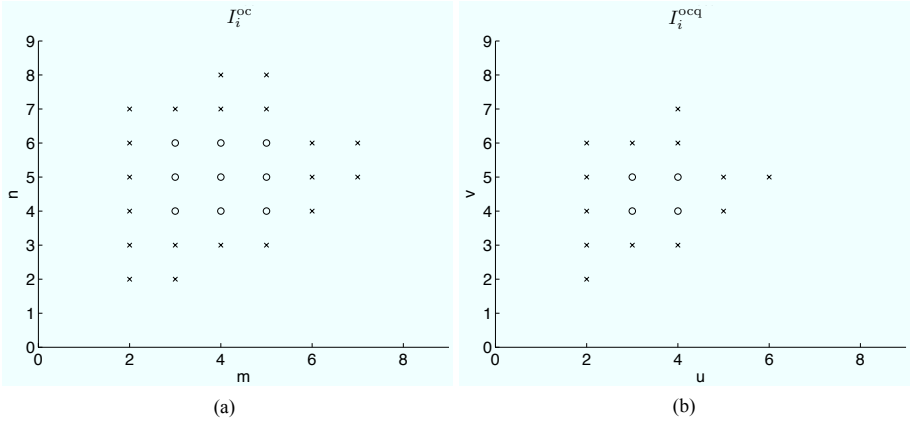


Figure 6.6: (a) I_i^{oc} where '$\times$' and '$\circ$' respectively represent a vertex of an intersection octant and internal octant in i-th octree slice. (b)I_i^{oc} where total number of points are reduced from 29 to 17.

The value of I_0^{oc} at point (m, n) is then given by

$$I_i^{oc}(m,n) = obj(x,y,z)\delta\left(x - mt, y - nt, z - it\right), \tag{6.3}$$

where m, n, and $i \in \mathbb{Z}$, and $\delta(\cdot)$ is a 3D delta function, which has a samplling interval[4] t.

An interesting observation of a sliced octree data is that every four nonzero points, forming the 4-connected neighbours in I_i^{oc}, are from the same octant. To treat these points equally and reduce the number of processing data, I_i^{oc} is transformed to a quantised octree slice, i.e.,

$$I_i^{ocq}[u,v] = \prod_{j,k=0}^{1} I_i^{oc}(u+j, v+k). \tag{6.4}$$

An example of an octree slice is illustrated in Figure 6.6(a), where vertices of intersection octants (marked as $\times$) and nine vertices of four internal octants (marked as $\circ$) are represented by 29 points in I_i^{oc}. Since every four points belong to the identical octant, a quantised octree slice I_i^{ocq} removes redundant vertices so that a nonzero point in I_i^{ocq} represents four vertices from the same octant [see Figure 6.6(b)].

On the other hand, to represent MC data by the same slice index even though its sampling period is half of t, a binary image plane $S_i^{mcq}[u,v]$ is only set to 1 when a MC vertex is found within $ut \le x < (u+1)t$, $vt \le y < (v+1)t$ and $it \le z < (i+1)t$.

[4]In the experiments, t is set to w, the size of an octant.

Thus, the volume of an octant associated with four nonzero points in I_i^{ocq} is t^3, whilst the volume of nonzero points at $[u, v]$ in the quantised MC slice is bounded by

$$\frac{T^3}{8} \leq vol(I_i^{mcq}[u, v]) \leq \frac{7T^3}{8}, \tag{6.5}$$

where $vol(\cdot)$ is a function which estimates the volume of nonzero points in the quantised MC slice. The actual volume of an object v_{obj} is smaller than the volume of MC slices, and which is smaller than the volume of the octree slices, i.e.,

$$\begin{aligned} v_{obj} < \sum_{i,u,v} \big\{ & vol(I_i^{mcq}[u, v]) + t^3 \left(I_i^{ocq}[u, v] \right. \\ & \left. - \; I_i^{mcq}[u, v] I_i^{ocq}[u, v] \right) \big\} < \sum_{i,u,v} \left\{ I_i^{ocq}[u, v] t^3 \right\}. \end{aligned} \tag{6.6}$$

In practice, however, the volume of MC slices often violate (6.6) because the erroneous classification of octree vertices fails to correctly locate MC vertices.

Another observation from a sliced octree data is that each quantised octree slice of a non-convex object can have multiple clusters that are linked 8-neighbouring points on the quantised octree slice. These multiple clusters need to be connected to other clusters in adjacent slices to define a local convexity. The clustering in a quantised octree slice is trivial because points belonging to the same cluster are conglomerated in accordance with the presence of internal octants. Thus, identifying a cluster in I_i^{ocq} is simply a search for connected nonzero points among eight neighbours. However, clustering in I^{mcq} is not similarly straightforward. A decision on the clustering and connecting of clusters in I_i^{mcq} is based on the Bayesian decision making rule [104] and *a priori* information of the decision is obtained from I_i^{ocq}.

Four examples of imperfect silhouettes which are obtained by simple thresholding are shown in the second row of Figure 6.7(a) with the corresponding actual images shown in the first row. An octree of a dummy from sixty silhouettes is illustrated in Figure 6.7(b). The octree is obtained from a seven-level of octree construction from a 40[cm]x40[cm]x40[cm] initial octant, i.e., the smallest size of octant is 0.625[cm]. When the traditional MC is applied to Figure 6.7(b) it produces holes on the resultant surface in addition to unattached segments. furthermore, some holes still remain in the VMC with

85% voting threshold, which produces the best surface as shown in Figure 6.7(c). The proposed method connects imperfect surface (e.g., holes) using the connectivity information obtained from Figure 6.7(b).

A total of 29 quantised octree slices and 60 clusters are found in Figure 6.7(b). Some slices of the octree, I_i^{ocq}, are illustrated in Figure 6.8 where each nonzero pixel in a slice indicates an octant, and octants belonging to the same cluster identification (ID) have identical grey value. For example, the slice 11 is for $z = 6.875$[cm], which is at shoulder height of the dummy.

6.4.2 Identifying a local convexity

A local convexity is identified by two processes: clustering on I^{mcq} and connecting clusters between slices. Given a cluster conditional pdf $p(\vec{t}|c_i)$ which gives the probability of a test data $\vec{t}$ belonging to a class c_i, the problem of clustering is solved by searching for the maximum probability.

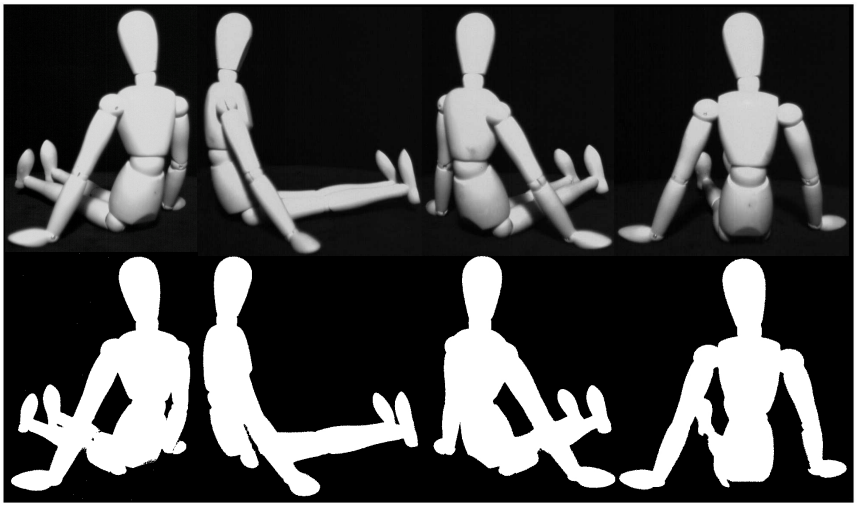
A probability mass function of j-th cluster in i-th quantised octree slice $P_i(c_j)$ is defined by the number of j-th cluster data n_j and the total number of nonzero points n_k, i.e.,

$$P_i(c_j) = \frac{n_j}{\sum_{k=0} n_k}. \tag{6.7}$$

Thus, the sum of cluster probabilities in a slice, i.e., $\sum_{j=0} P_i(c_j)$, is 1. From this *a priori* knowledge a cluster ID is predicted when a nonzero point is found in I_i^{mcq}. For example, if two clusters are found in I_i^{ocq} and their probability are $P_i(c_0) = 0.7$ and $P_i(c_1) = 0.3$ then a nonzero point in I_i^{mcq} is classified as cluster c_0.

If points in j-th cluster are known to be more likely to be in a certain part of a slice, then the first a priori knowledge is enhanced by combining it with a second *a priori* knowledge obtained from the distribution of cluster data. If $\vec{t} = [u \quad v]^{\mathrm{T}}$ represents the 2D position of a clustered data in i-th slice, a joint pdf of two random variables $\vec{t}$ and c_j that are not independent is

$$p_i(\vec{t}, c_j) = p_i(\vec{t}|c_j)P_i(c_j) = p_i(c_j|\vec{t})P_i(\vec{t}). \tag{6.8}$$



(a)

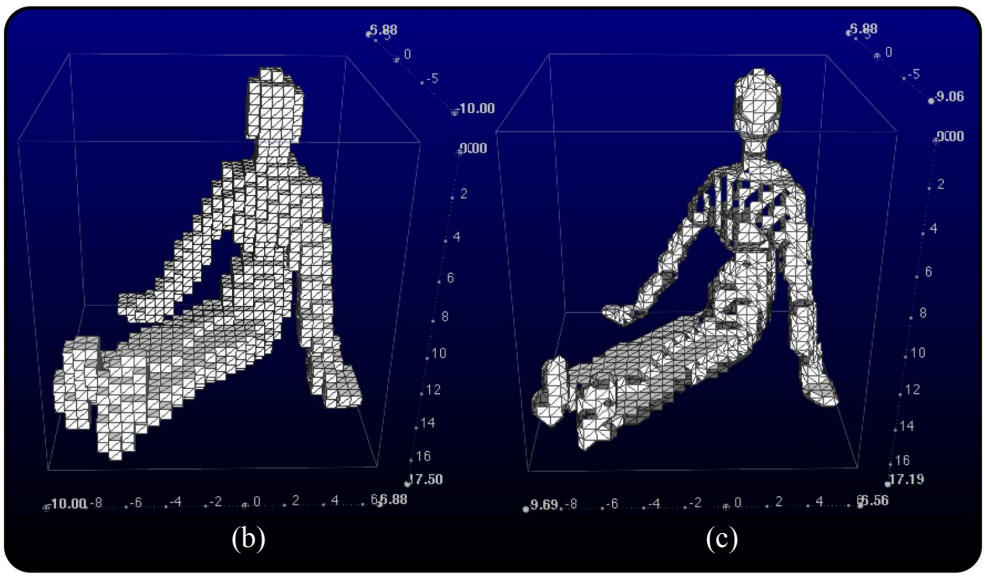


Figure 6.7: (a) Examples of silhouettes obtained using simple thresholding, which produces imperfect occluding contours. (b) A seven-level of octree from sixty silhouette images. (c) The best surface result from VMC with 85% voting threshold. The unit of three axes in (b) and (c) is [cm].

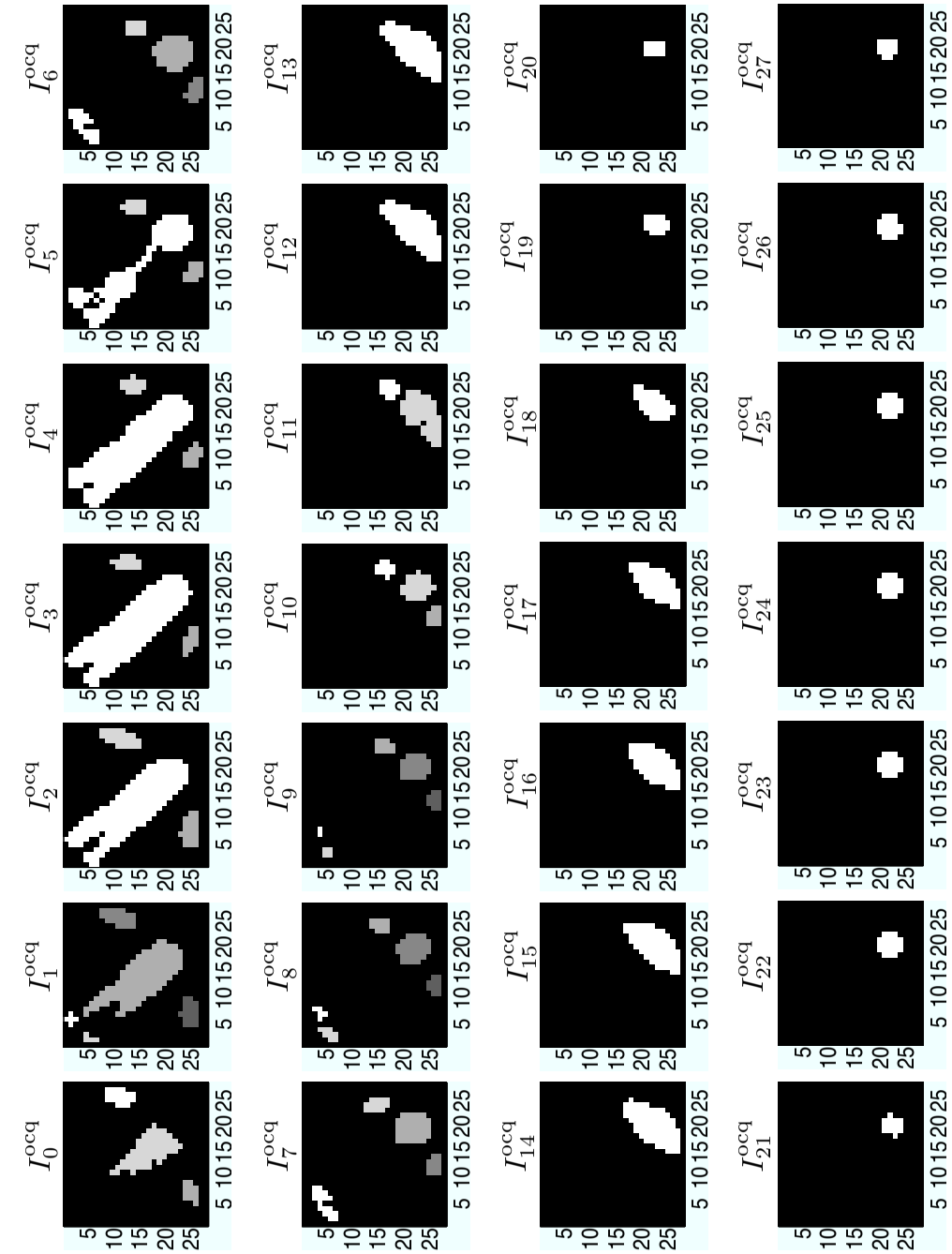


Figure 6.8: Examples of slices I_i^{ocq} where each slice is quantised as 28×28 grid, where a nonzero value on a slice represents an octant.

The *a priori* knowledge in (6.8) is converted to a test data conditional pdf of a cluster $p_i(c_j|\vec{t})$ called a posteriori pdf,

$$p_i(c_j|\vec{t}) = \frac{p_i(\vec{t}|c_j)P_i(c_j)}{\sum_j p_i(\vec{t}|c_j)P_i(c_j)}. \tag{6.9}$$

Therefore, in order to cluster MC vertices based on the Bayesian rule, $p_i(\vec{t}|c_j)$ in (6.9) needs to be estimated.

Without any assumption on $p_i(\vec{t}|c_j)$, a cluster conditional pdf of a test data $\vec{t} = [u \quad v]$ is estimated using the Parzen non-parametric density estimator [104] from I_i^{ocq}. If the density function is known, the probability P that j-th cluster data is found in a square with area w_n centred at (x, y) is

$$P = \sum_{v=y-w_n/2}^{y+w_n/2} \sum_{u=x-w_n/2}^{x+w_n/2} p_i(\vec{t}|c_j). \tag{6.10}$$

In practice, the only information available is that n_j data are classified as j-th cluster in I_i^{ocq}. Thus, the probability that k data fall within the square is $P \simeq k/n_j$, and the ratio k/n_j converges to the true P as the number of samples is increased.

The Parzen estimator increases the number of samples in a square by interpolating between samples in a window, i.e.,

$$p_i(\vec{t}|c_j) = \frac{1}{n_j} \sum_{i=0}^{n_j} w(\vec{t} - \vec{t}_i), \tag{6.11}$$

where $w(\cdot)$ is the Gaussian window with an area equal to w_n. Since the size of a Gaussian window confines the range of the Gaussian function, the degree of smoothness of the pdf is determined by w_n.

Each I_i^{ocq} is used to estimate cluster conditional pdfs and those in the same slice are stored in a pdf cube. Therefore, a cluster decision function $d(\vec{t})$ which determines a cluster ID of a test data $\vec{t} = [u \quad v]^{\text{T}}$ in the i-th quantised MC slice is

$$d(\vec{t}) = \max_j \left\{ p_i(\vec{t}|c_j)P_i(c_j) \right\}, \tag{6.12}$$

where j represents a cluster ID in the i-th slice. Figure 6.9 illustrates the construction of

pdf cubes using (6.10) from the slice 0 to slice 19 shown in Figure 6.8.

Note that the cluster conditional pdf's of connected clusters are similar because the second property assumes that the connected clusters have a similar shape. Thus, a decision function for connecting a cluster c_{id} on the current slice i needs to refer to the pdf cube in the next slice, i.e.,

$$e(c_{id}, i) = \max_j \{ \sum_{\vec{t} \in c_{id}} p_{i+1}(\vec{t}|c_j) P_{i+1}(c_j) \}. \tag{6.13}$$

The connection between clusters is summarised in a connection tree, where a node of the tree represents a cluster and it stores information of bidirectional connection, i.e., a connected tail ID and head cluster ID. A part of the connection tree of the octree is shown in Table 6.2. Each row of the tree table shows a cluster ID (CID), the slice number (SNO), head and tail cluster ID (HID, TID). x indicates that there are no connections, e.g., if HID is 'x' then a new local convexity starts from the current slice. On the other hand, a local convexity terminates the connection if a TID is x. Since it is assumed that there are no unattached clusters in an object, a cluster should not have x as both of its head and tail ID. A slice with multiple clusters indicates that the object has non-convex shape and the resulting tree table has multiple HID's or TID's. To show similarity of connected clusters, a Cr column is added in the table. It is defined by the modulus of the approximated correlation coefficient between CID and TID's. For example, the Cr of two clusters c_m and c_n in adjacent slices is estimated by

$$g(c_m, c_n) = \left| \frac{(\sum_i p_j(\vec{t}_i|c_m) p_{j+1}(\vec{t}_i|c_n))^2}{\sum_i p_{j+1}^2(\vec{t}_i|c_n) \sum_i p_j^2(\vec{t}_i|c_m)} \right|, \tag{6.14}$$

where c_m is the m-th cluster in the current slice and c_n is the n-th cluster, which is found as the TID of c_m in the next slice. When a current cluster has multiple tails then an average Cr value is used.

6.4.3 Local surface construction

A local convexity is defined by two connected clusters in adjacent slices. If the data is sliced reasonably small and every slice has a single cluster, the 3D hull algorithm will

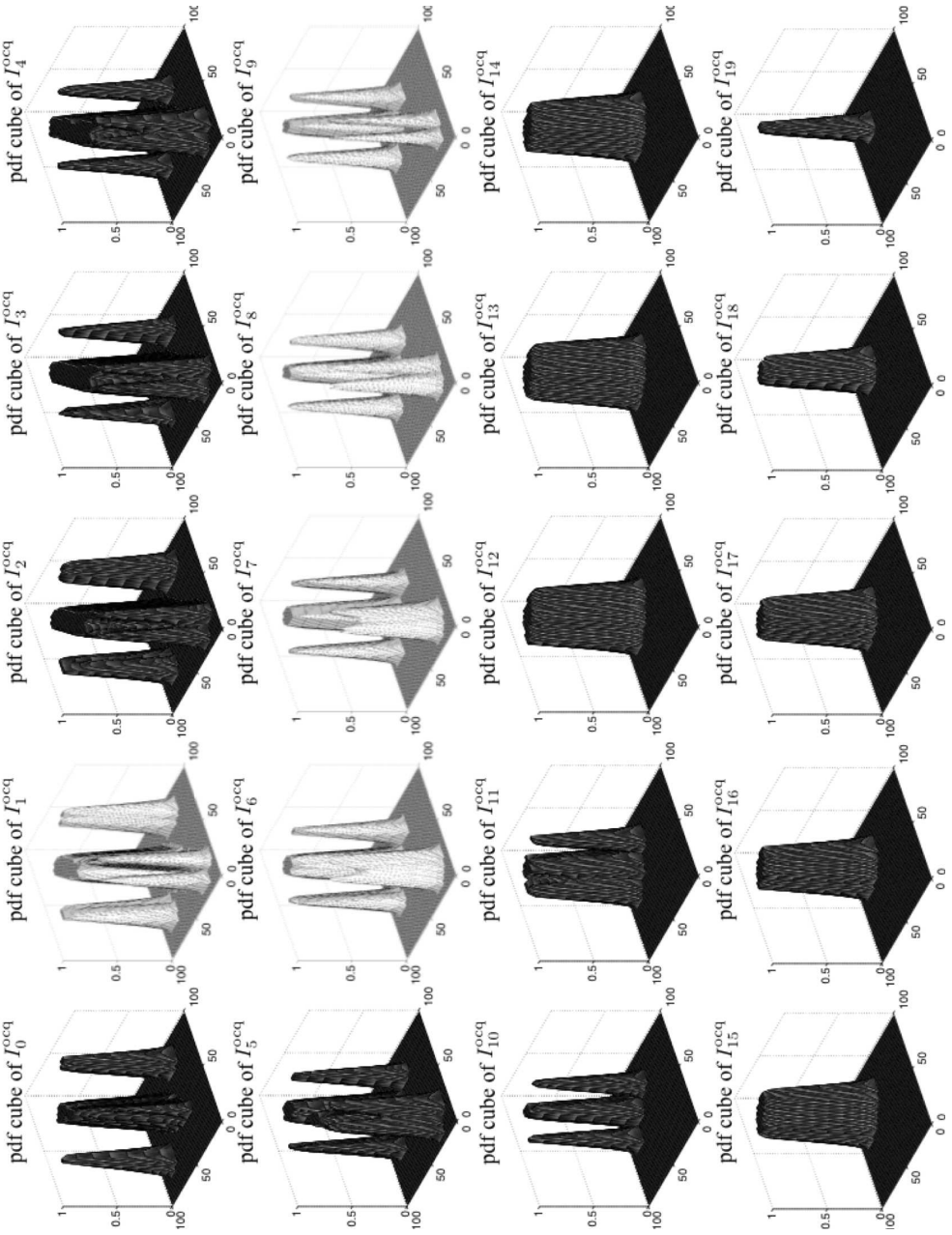


Figure 6.9: Examples of pdf cubes: a 3D pdf cube contains every cluster conditional pdf found in a quantised octree slice. A cluster conditional pdf interpolates 110x110 pixels and the size of the Gaussian window is 3.

Table 6.2: Connection tree.

SNO[a]	CID	HID	TID	Cr	SNO	CID	HID	TID	Cr
0	0	x	6	0.961531	8	30	25	35	0.857974
0	1	x	5	0.68902	8	31	26	36	0.935118
0	2	x	7	0.908821	8	32	27	37	0.753226
1	3	x	8	0.0303864	9	33	28	x	
1	4	x	8	0.0275733	9	34	29	x	
1	5	1	8	0.788193	9	35	30	38	0.955456
1	6	0	9	0.97162	9	36	31	39	0.961278
1	7	2	10	0.9357	9	37	32	40	0.808304
2	8	5, 3, 4	11	0.976206	10	38	35	41	0.809127
2	9	6	12	0.685834	10	39	36	42	0.648546
2	10	7	13	0.596339	10	40	37	42	0.195431
3	11	8	14	0.987193	11	41	38	43	0.158091
3	12	9	15	0.670017	11	42	39, 40	43	0.766806
3	13	10	16	0.786933	12	43	42, 41	44	0.995385
4	14	11	17	0.732324	13	44	43	45	0.978099
4	15	12	18	0.946549	14	45	44	46	0.97904
4	16	13	19	0.862888	15	46	45	47	0.971152
5	17	14	22, 20	0.39/0.176[b]	16	47	46	48	0.986242
5	18	15	21	0.982989	17	48	47	49	0.859554
5	19	16	23	0.927811	18	49	48	50	0.724647
6	20	17	24	0.951042	19	50	49	51	0.831379
6	21	18	25	0.921267	20	51	50	52	0.971805
6	22	17	26	0.963373	21	52	51	53	0.89483
6	23	19	27	0.918805	22	53	52	54	1
7	24	20	29, 28	0.46/0.46	23	54	53	55	1
7	25	21	30	0.864548	24	55	54	56	1
7	26	22	31	0.962148	25	56	55	57	0.990933
7	27	23	32	0.980496	26	57	56	58	0.899973
8	28	24	33	0.87784	27	58	57	59	0.753494
8	29	24	34	0.799906	28	59	58	x	

[a]Keys: Slice Number (SNO), Current cluster ID (CID), Head cluster ID (HID), Tail cluster ID (TID), Correlation distance (Cr).

[b]Cr between CID and the second TID

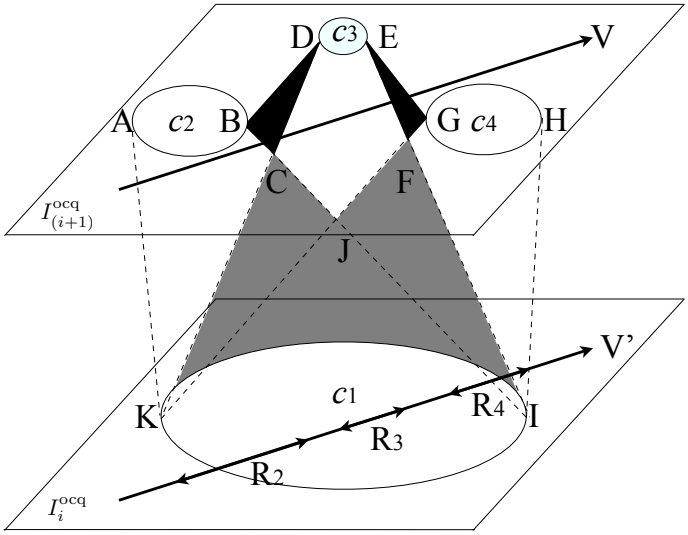


Figure 6.10: 1:n branching case. If the 3D hull algorithm is simply applied to multiple connections, some object details will be smoothed. To avoid the smoothing, the cluster c_1 is divided into 3 subregions, R_2, R_3 and R_4 on the projection of the eigen vector V' and n 1:1 connections are made.

construct a good surface of a local convexity because the connected clusters are regarded as convex. However, if an object is not convex, a local convexity can have multiple connections (see Figure 6.10). This means that the local convexity does not correspond to a convex shape since a convex shape is only possible with 1:1 cluster connection. Thus, the 3D hull algorithm will smooth some details of the object.

Figure 6.10 shows an example of multiple connections. A cluster c_1 in slice I_i^{ocq} is connected to three clusters c_2, c_3 and c_4 in slice $I_{(i+1)}^{\text{ocq}}$, and this make a 1:n branching connection. An opposite case is a n:1 merging connection which is found when n clusters are merged to a cluster in the next slice (see CID 8 in Table 6.2). These multiple connections are normally found in the octree contruction of a non-convex object. If a cluster is branching to several clusters in an arbitrary distant position, unexpected connections created by the 3D hull algorithm expand the surface construction. For example, the application of the algorithm to the multiple connections in Figure 6.10 results in a local surface connecting A, B, D, E, G, H, I and K. This causes black areas to be added and they smooth some details of the object.

However, if the multiple connections are simply treated as multiple 1:1 cases, then connections between c_1 and c_2, between c_1 and c_3, and between c_1 and c_4 unnecessarily

duplicate the surface in the grey area enclosed by K, C, J, F and I in Figure 6.10. To address this problem the proposed method divides multiple connections into n 1:1 connections with an appropriate division so as to minimise possible duplication of surface patches in the common area. For example, c_1 in Figure 6.10 is divided into 3 subclusters to make three 1:1 connections.

The division is performed along the best representative vector of the multiple clusters which is estimated by an eigen analysis. If $\vec{t}_{ij} = [u \quad v]^{\mathrm{T}}$ represents the i-th nonzero point in cluster j, m data points in the n-th branched cluster are $X = \begin{bmatrix} \vec{t}_{11} & \vec{t}_{21} & \cdots & \vec{t}_{mn} \end{bmatrix}$ and the covariance matrix C of the data is

$$C = \sum_j \sum_i \left(\vec{t}_{ij} - \vec{m}\right)\left(\vec{t}_{ij} - \vec{m}\right)^{\mathrm{T}}, \tag{6.15}$$

where $\vec{m}$ is the mean of X. If there are orthonormal column vectors $\vec{e}_i$, the projection of the matrix C onto the orthonormal vectors is

$$C[\vec{e}_1\vec{e}_2] = [\vec{e}_1\vec{e}_2]\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \tag{6.16}$$

where λ_i is the eigen value of the eigen vector $\vec{e}_i$ of C. X is also represented by a weighted sum of the eigen vectors, i.e., $X = [\vec{e_1}\vec{e_2}]\left([\vec{e_1}\vec{e_2}]^{\mathrm{T}}X\right) = [\vec{e_1}\vec{e_2}]X'$. These eigen vectors are used as orthonormal basis to express $\vec{t}_{ij}$, and the eigen vector corresponding to the maximum eigen value of C is the best vector which represents X, e.g., if $\lambda_1 > \lambda_2$ then $\vec{e}_1$ is the best vector.

Once the best eigen vector of the multiple clusters is found, each column vector of X is projected onto $\vec{e}_1$ to find the distribution of clusters on the eigen vector, i.e., a location of $\vec{t}_{ij}$ on the eigen vector $\vec{e}_1$ is $l_{ij} = \vec{t}_{ij}^{\mathrm{T}}\vec{e}_1$. Thus, the minimum and maximum locations of j-th cluster, $(l_{\min}, l_{\max})_j$, indicate the distribution of j-th cluster. To divide a cluster into multiple subclusters, the data distributions are normalised. Finally, the cluster to be divided is projected onto $\vec{e}_1$ and divided according to the normalised cluster distributions. In Figure 6.10, the eigen vector of the three clusters are represented as V in $I^{\mathrm{ocq}}_{(i+1)}$. The projection of V onto $I^{\mathrm{ocq}}_{(}i)$, V', is used to divide the three clusters and the dividing ranges are denoted by R_2, R_3 and R_4.

```
1  //1. Create pdf cubes from octree data
   //2. Make a connection tree from pdf cubes
3  //3. Classify MC vertices using pdf cubes
   //4. Estimate local surface
5  for i = 0 until end of slices
       for j=0 until end of clusters
7          // surface between the current slice and next slice
           if (TailIDs.size() > 1)      // 1:n branching connection
9              collect MC vertices having index which forms a 1:n local convexity
               call function, MakeLocalFace
11         else if (TailIDs.size() == 1)     // 1:1 connection
               collect MC vertices having index which forms a 1:1 local convexity
13             call function, MakeLocalFace
           end if
15
           // surface between the current slice and previous slice
17         if (HeadIDs.size() > 1 && i > 0)        // n:1 merging connection
               collect MC vertices having index which forms a n:1 local convexity
19             call function, MakeLocalFace
           end if
21     end for
   end for
```

Figure 6.11: Surface generation.

6.4.4 Implementation

A pseudo code for surface construction algorithm is shown in Figure 6.11 where the major surface construction routine are shown from line 4 to line 22 in addition to the initial preparations of pdf cubes and connection tree in the first three lines. User-defined CPdfCube class is designed in this thesis to encapsulate all functions regarding data slicing, quantising the slices, constructing pdf cubes, classifying MC vertex, and estimating a connection tree. Each cluster on a slice defines local convexity in the previous (line 17-20) and next slices (line 8-14), and a function MakeLocalFace collects MC vertices belonging to the local convexity followed by the convex hull algorithm. Merging and branching cases are taken into consideration in the MakeLocalFace function, i.e., it divides clusters based on the data distribution before the local convex hull algorithm is applied.

6.5 Experimental results

This section demonstrates surface construction results according to various noise conditions. Experiments includes:

- estimation of the quality of traditional surface construction (e.g., MC and VMC) when noise is added in the silhouette images. The results are quantified in terms of the lost octants ratios, i.e. low ratio is better;
- comparing the surface construction of traditional 3D convex hull with the proposed LCH method using four objects with different shape complexity;

- finding the number of surface triangles obtained from six methods (i.e., MC, VMC, DT, CH, LDT and LCH) relative to the noise level;
- demonstrating the computing complexity of the proposed method and comparing it with other methods;
- surface reconstruction using images of four objects, which involve practical noise (i.e, noise is not synthesised in this test).

The proposed algorithm has been evaluated on four objects with shapes of different complexities as shown in the first row of Figure 6.12. The oil burner [Figure 6.12(a)] has four non-convex details in its sides. The dragon [Figure 6.12(b)] has a more complex shape than Figure 6.12(a) with numerous merged or branching clusters in its slices. The bust [Figure 6.12(c)] has only one cluster in every slice but it is not convex. Finally, the vase has the simplest shape.

Each image of an object is captured as a 640×480 colour image and from the 60 images of each test object, an eight-level octree[5] was constructed (see second row of Figure 6.12). For an appropriate silhouette detection, thresholding is performed after Gaussian smoothing and contrast enhancement. A seed-fill operation [24] is then applied to minimise silhouette detection errors. When an object has non-convex details as shown in Figure 6.12(a) and (b), any remaining errors are manually removed after the seed-fill operation. A projection matrix at the reference position is estimated by a linear SVD method from a 3D calibration rig with 2.45438[px] re-projection error.

The MC surfaces of the four test objects are shown in the last row of Figure 6.12. Since the MC results are obtained from small silhouette and projection error, MC triangles are correctly constructed from most intersection octants. However, some sharp details, e.g., tail of the dragon, are partly removed because of the resolution of the octant, i.e., the tail of the dragon is too sharp when compared to the current octree resolution, resulting in case b or c intersection.

To simulate images affected by erroneous silhouettes and calibration error, salt and pepper noise is added to the silhouette images. The added noise ratio is defined as

[5]Higher level of octree than results in Chapter 2 reduces the chance of losing surface triangles from MC, and make the connectivity and continuity assumptions more reliable

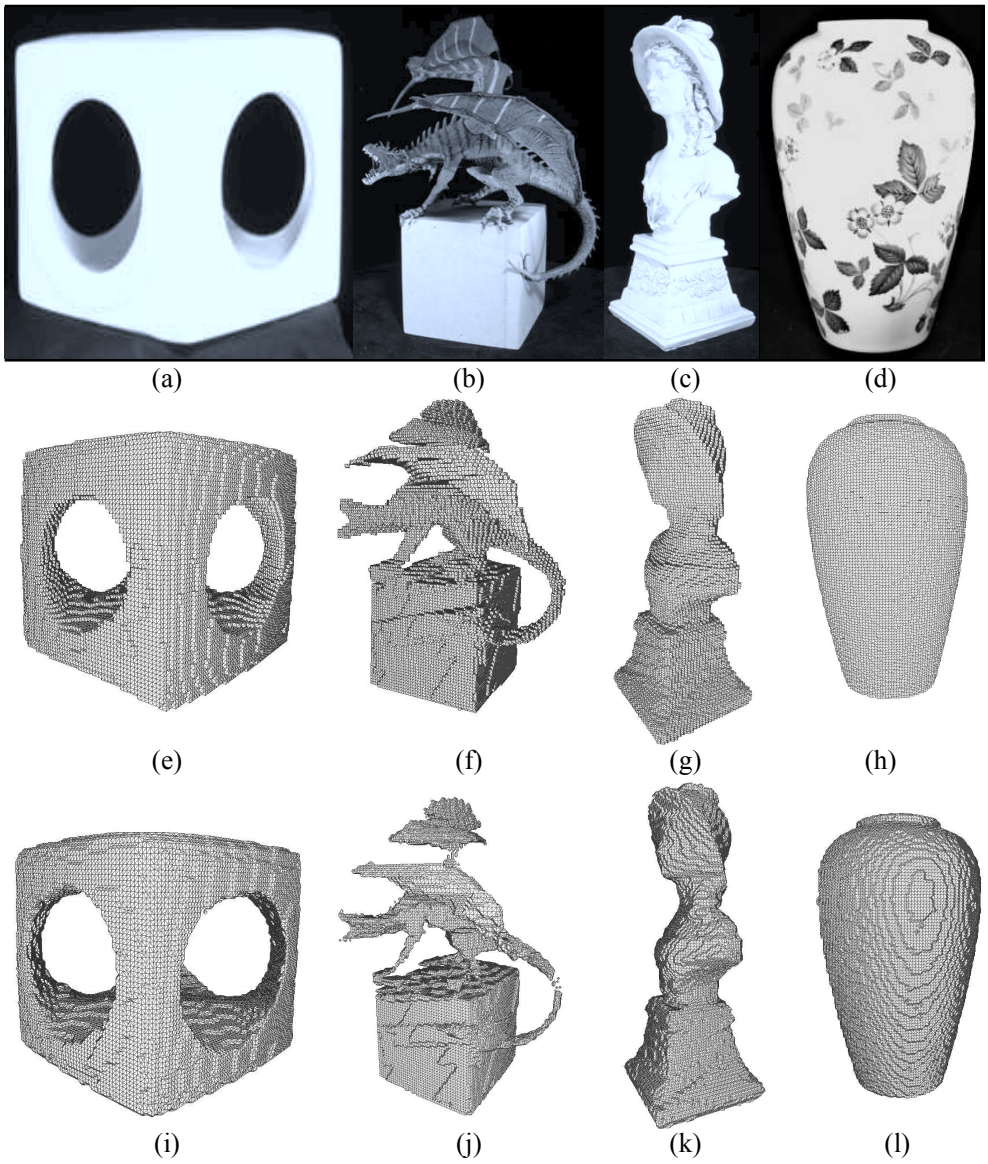


Figure 6.12: Eight-level octrees and MC surfaces of four test objects: (a)-(d) Images of objects at the reference position; (e)-(h) The corresponding octrees respectively with 362320, 75504, 267072 and 378448 octants; (i)-(l) MC surfaces from (e)-(h).

the ratio of the number of contaminated pixels to the total number of pixels, and the positions of the noise are selected by a uniform random distribution. Figure 6.13(a)-(d) show examples of silhouette images with 10% noise added. Since MC is sensitive to noise, i.e., as the noise ratio increases it fails to construct surface triangles from more octants. Figure 6.14(a) shows the lost octants ratio for varying noise ratios, where the lost octants ratio is defined as the ratio of the number of lost octants that do not result in a surface to the total number of octants. When there is no noise the lost octants ratio is almost nil. However, when a small amount of noise is added to the silhouette images, a significant number of octants do not result in a surface. Examples of MC surfaces when 5% noise is added are shown in Figure 6.13(e)-(h).

VMC can address the noise sensitivity of MC. It can reduce the number of lost octants with an appropriate voting threshold. Figure 6.14(b) shows the lost octants ratio of two data sets containing the four test objects. The first set is corrupted with 10% noise and the VMC results are denoted by solid lines. The second set is corrupted with 15% noise and the VMC results are denoted by dashed lines. The voting threshold which gives the best performance is referred to as the best voting threshold. The best voting threshold for the first set is about 90%, but when more noise is added a smaller voting threshold becomes appropriate to minimise the lost octants ratio. However, too small a voting threshold increases the lost octants ratio. This is because as the voting threshold becomes smaller it is possible to have an intersection octant with 8 inside corners. This particular case is regarded as an inside octant by MC and no surface is constructed.

The best VMC results from images with 10% noise added are shown in Figure 6.13(i)-(l). As illustrated, VMC can minimise the lost octants but the results are degraded when compared to the ideal MC results, e.g., the surfaces are not continuous. Thus, the best VMC result may include sufficient surface vertices but the surface connection is not always correct.

When a simple convex hull (CH) algorithm is applied to the best VMC result, many shape details are lost [see Figure 6.15(a)-(d)] whilst the results of applying the proposed method shown in Figure 6.15(e)-(g) preserve them. However, when the object is totally convex [Figure 6.15(d)], CH is the best in terms of processing time, memory usage and approximation efficiency. To show the approximation efficiency for varying

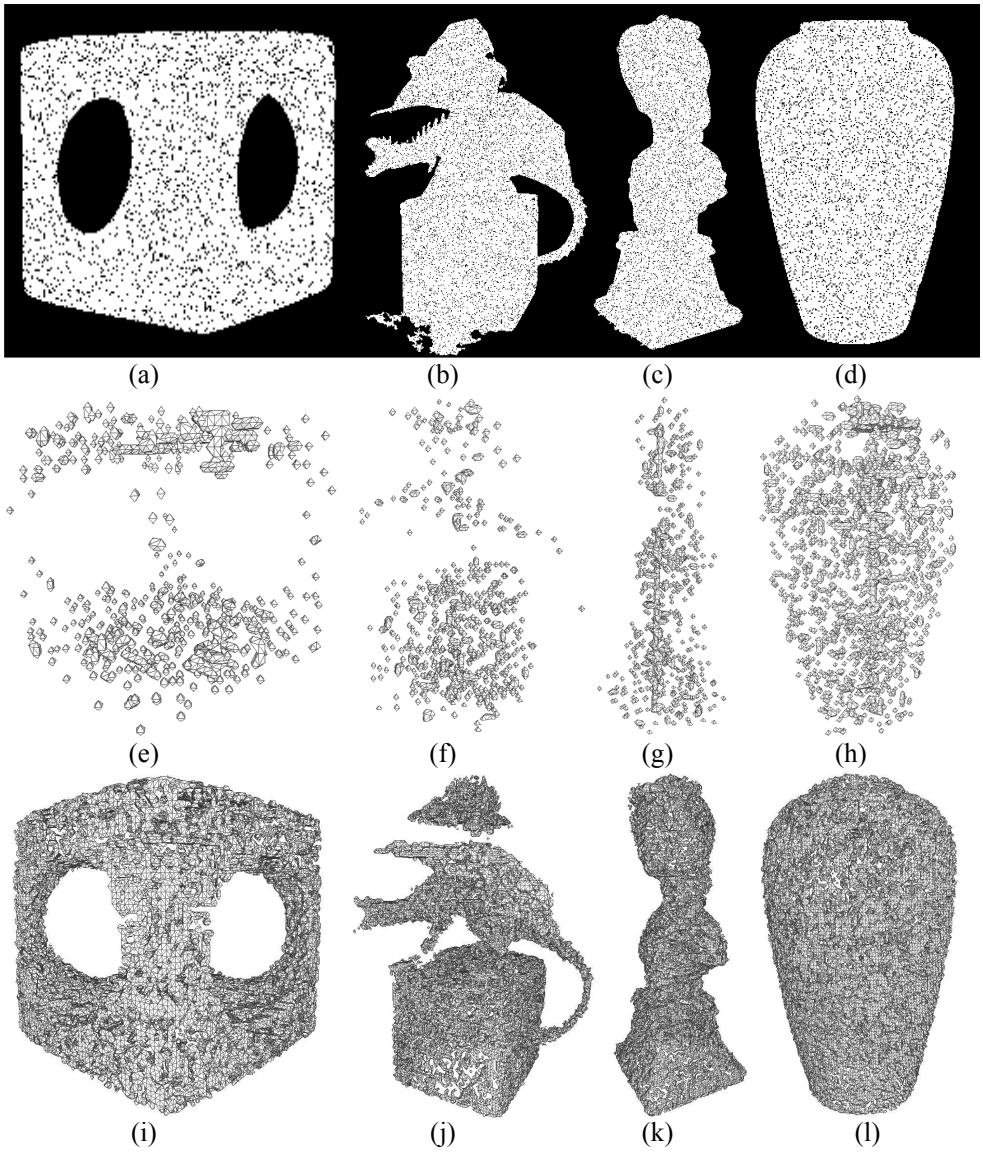


Figure 6.13: (a)-(d) Silhouette images with 10% noise added; (e)-(f) MC surfaces estimated from silhouette images with 5% noise added; (i)-(l) the best VMC results from silhouette images with 10% noise added.

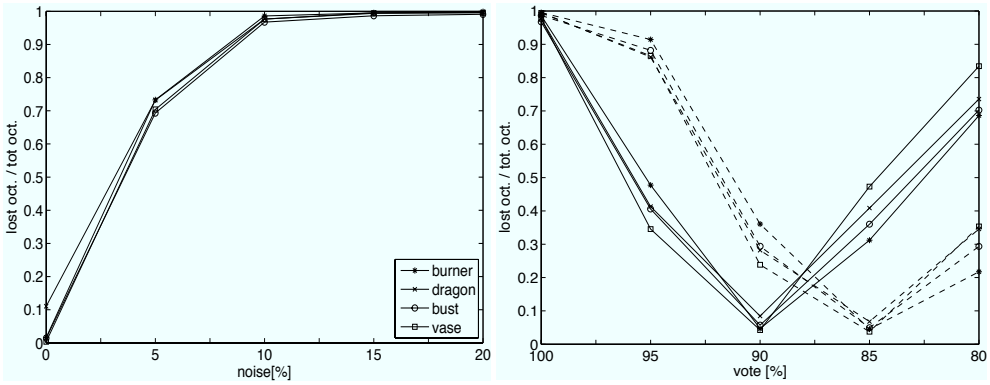


Figure 6.14: (a) Lost octants ratios using MC for varying noise ratios. (b) Lost octants ratios using VMC for varying voting thresholds.

Table 6.3: Number of surface triangles on reconstructed burner for varying noise ratios.

	0%	**5%**	**10%**	**15%**	**20%**
MC	14271	5640	296	124	24
VMC[a]	14271	43069	44489	44622	45520
DT	160048	541476	575812	58540	601980
CH	410	434	388	430	410
LDT	245112	850428	915312	918676	949672
LCH	3662	4380	4566	4538	4426

[a]obtained from the best VMC with voting threshold from 100% to 80%

noise ratios, the total number of surface triangles on the reconstructed burner using 6 methods are summarised in Table 6.3, where LDT and LCH represent local DT and local CH, respectively. LDT which uses DT algorithm for constructing local convex hull is developed to show the efficiency of the proposed method, LCH.

Since DT, CH, LDT and LCH construct the surface from the best VMC result, they have the same noise robustness as the best VMC. Thus, the number of surface triangles constructed by these methods does not change significantly even after noise addition. CH approximates shape with the smallest number of triangles but the quality of the visual appearance of the result depends on the shape of objects. DT constructs similar surfaces to CH but the number of triangles are significantly increased because DT forms tetrahedrons from every four 3D points. Therefore, LDT results in the largest number of triangles because it performs DT between every two slices. On the other hand,

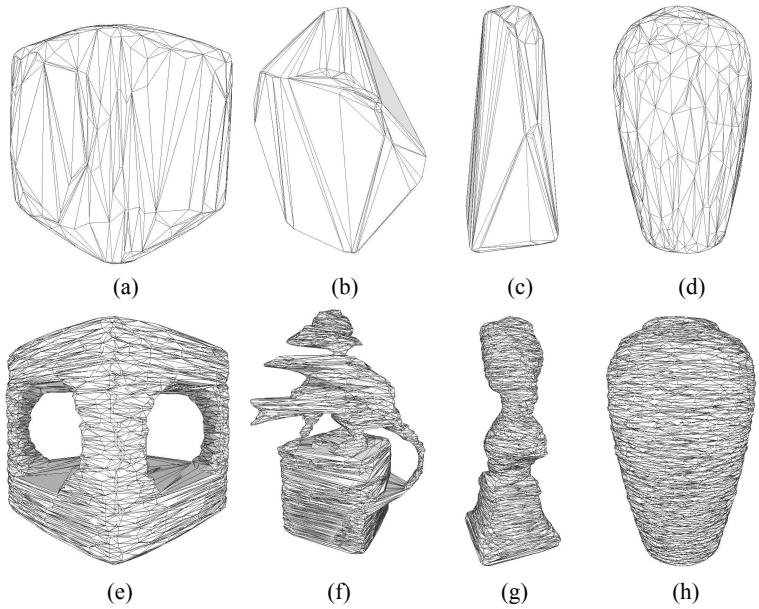


Figure 6.15: Surface reconstruction from the best VMC result: (a)-(d) using the convex hull algorithm; (e)-(h): using the proposed method.

LCH is the second most efficient method, but unlike CH it is able to construct non-convex details.

The performance of the proposed method (LCH) in terms of the required CPU time and peak memory usage is compared with 5 surface construction algorithms: MC, VMC, CH, DT and LDT. Qhull code is used for the implementation of CH and DT [105], and LCH and LDT also incorporate Qhull code. A look up table for the 256 possible cases of surface construction [12] is provided for MC to avoid rotational and complementary symmetry checking, which is known to be the most time consuming process in MC. In general, CH requires the shortest time, followed by DT, LCH, LDT, VMC and MC [see Figure 6.16(a)]. In the case where DT constructs a large number of tetrahedrons for the vase, DT is slower than LCH.

However, the proposed method uses more memory to store data slices, tree tables and pdf information. Thus, the peak memory usage of LDT and LCH is 3 to 6 times higher than general CH and DT [see Figure 6.16(b)]. The order of complexity of DT used in the test is reported to be $O(n \ln v)$ [105] and MC is $O(n)$, where n is the number of input points and v is the number of output vertices. However, the order of the proposed

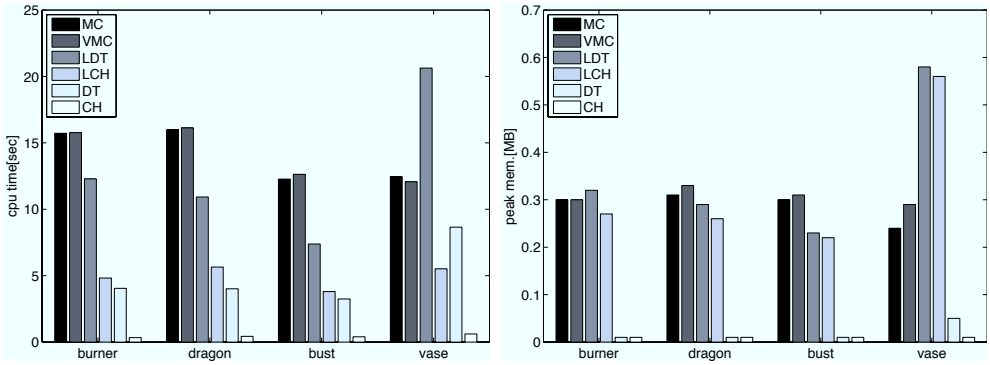


Figure 6.16: (a) CPU time and (b) peak memory usage required by 6 algorithms to construct 4 objects.

method is $O(m \times k)$, where m is the number of slices and k is the number of clusters. The proposed method also needs to incorporate the Qhull algorithm with complexity $O(n \ln v)$ [64].

The last experiment is performed on a dummy which can move its limbs and makes arbitrary non-convex shapes. Some examples of the 60 input images used are shown in Figure 6.17(a)-(c). When only a simple thresholding process is applied, the silhouette images include detection errors as illustrated in Figure 6.17(d)-(f). For examples, the right leg in Figure 6.17(d) has shrunk and the left arm in Figure 6.17(e) has unexpected noise because of shading effect. Furthermore, although the images of the right arm does not have significant noise, the contour of the arm are altered in all its silhouette images, and as a result most parts of the right arm vanish in its MC surface. Nonetheless, a seven-level octree with 30[cm] as the initial octant size [Figure 6.17(g)] results in an acceptable 3D reconstruction. But the results using MC and 90% VMC have many unattached segments and holes [see Figure 6.17(h) and (i)]. LCH produces the best continuous surface [Figure 6.17(j)].

Some additional reconstruction results are illustrated in Figure 6.18. The surface of a dummy shown in Figure 6.7 is illustrated in Figure 6.18 (a), and the other two objects are from Figure 2.10(b) and (c). Since the shape of two objects in Figure 6.18(b) and (c) is close to convex, MC performs better than VMC [compare results shown in the second (MC) and third column (VMC) of Figure 6.18(b) and (c)]. The best VMC results shown

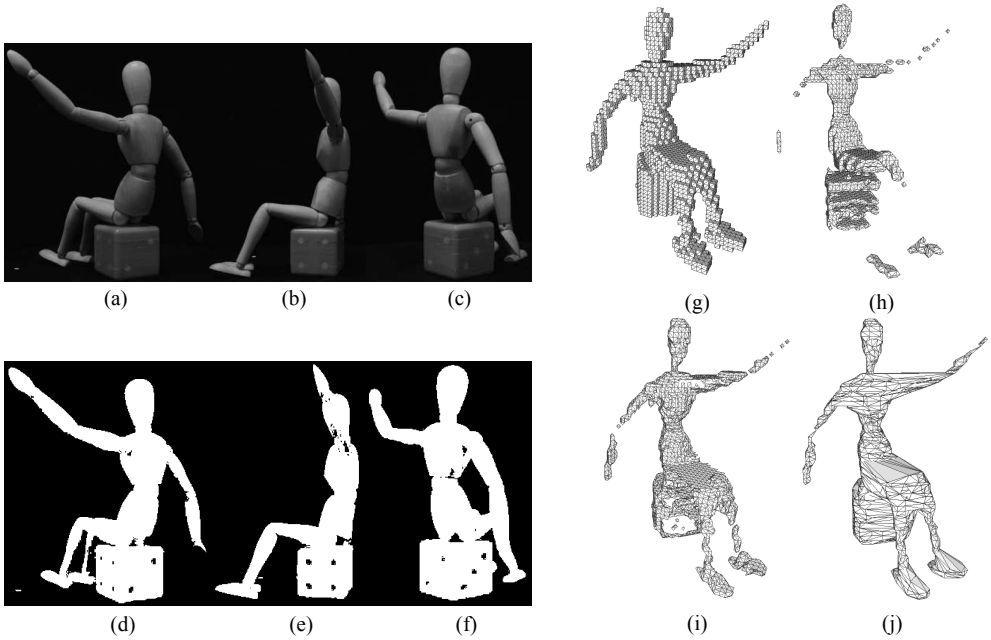


Figure 6.17: (a)-(c) A dummy with different poses. (d)-(f) The corresponding silhouette images using simple thresholding. (g) seven-level octree. (h) MC surface. (i) 90% VMC surface. (j) LCH surface.

in the third column of Figure 6.18 are obtained when voting thresholds are set to 95%, 85%, and 85%. The three LCH results shown in the last column are estimated from the VMC result of (a) and the MC results of object (b) and (c).

6.6 Conclusion

A surface constructed from 3D volumetric data facilitates the rendering of the object. This chapter explores a method which constructs triangular patches from an octree, and a robust construction is achieved by assuming two properties of a 3D object. The connectivity property presumes a surface covers all area of an object tightly without unattached object segments. The continuity property assumes an object as piecewise convex and a local convexity is similar in shape to the adjacent convexity if they are connected.

The proposed local hull-based surface construction (LCH) estimates a surface from local convexities. The best VMC result is used as its initial surface vertices, and slices are prepared from them. The sliced data is clustered based on a cluster conditional

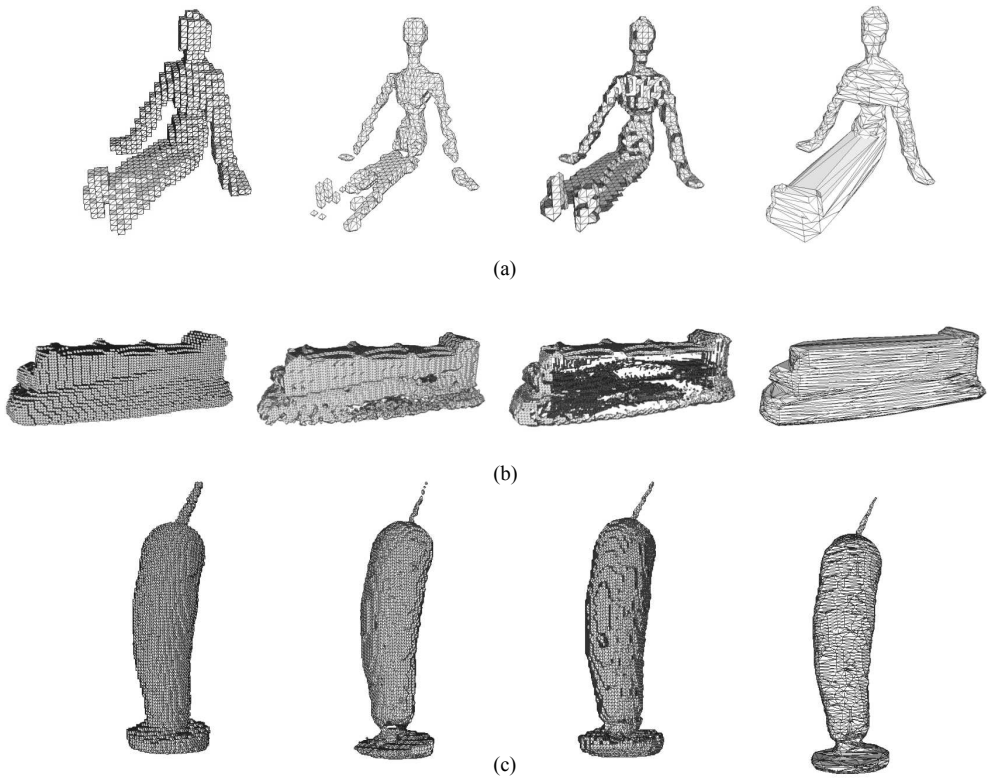


Figure 6.18: Octree, MC, VMC, and LCH results of (a) the dummy; (b) a school model and (c) courgette. The four objects are shown in Figure 2.10

pdf which is estimated from its octree, and the clusters in each slice are connected to its neighbouring clusters in adjacent slices by the Bayesian decision making rule in order to define local convexities. Finally, a convex hull algorithm creates local surfaces which are combined to complete the surface construction.

The experimental results show that LCH produces quality surfaces with good performance. Its approximation efficiency is better than those produced by other algorithms, e.g., MC, VMC, DT and LDT, requiring a reasonable CPU time. However, its peak memory usage is higher than CH and DT because the method needs to store local connection data. Also, any concavity in the xy plane may disappear in LCH, i.e., the concavity of a cluster in a slice is regarded as a 2D convex. This problem can be alleviated by dividing a 2D non-convex cluster into convex regions - it is like a 2D version of LCH. Another issue of the proposed method is the quality of the triangular patches, i.e., elongated or thin patches are caused by using a small slicing level and the convex hull

algorithm. However, it can be improved by inserting refining points into the side of the thin triangles.

Chapter 7

Conclusion and future work

7.1 Conclusions

This thesis presents a robust surface modelling method of a volumetric data reconstructed from multiple views. To achieve this goal, it investigates various computer vision algorithms in each chapter, e.g., robust image segmentation (Chapter 2), 3D reconstruction from multiple views (Chapter 2), non-linear model optimisation for a projection transform estimation (Chapter 3), 2D Delaunay graph for data clustering (Chapter 4), Hausdorff distance for cluster matching (Chapter 4), image descriptor for robust matching (Chapter 4), 2D convex hull algorithm (Chapter 5), the epipolar transfer in an image triplet (Chapter 5), MC, 3D convex hull, and 3D Delaunay for surface construction from volumetric data (Chapter 6). As a result of the research, this thesis proposes new approaches, which are required to accomplish the final goal, such as the projection matrix estimation from an approximate circular motion (Chapter 3), similarity invariant point cluster matching (Chapter 4), affine invariant clique descriptor matching (Chapter 4), unknown image calibration from point correspondences in an image triplet (Chapter 5), and 3D surface construction from non-convex volumetric data using the Bayesian decision making on sliced volumetric data (Chapter 6).

The volumetric reconstruction method presented in Chapter 2 approximates a 3D shape from multiple back-projection of silhouettes, and a result obtained by this approach is collectively referred to as SfS. Thus, the first process of SfS is the preparation of

silhouettes, which are produced by thresholding the background from an object image. For a more robust silhouette detection, the largest segment in the binary image is considered as a silhouette candidate, to which the seed-fill algorithm is applied to remove internal noise. As a volumetric representation, this thesis exploits an octree structure of voxels, and two octree construction algorithms are introduced in Chapter 2. The octree construction method I provides a straightforward implementation of an octree construction, i.e., a volumetric result is constructed by octants only from the last generation. Thus, this method is fast when an object has complex shape. Method II is more memory efficient and faster when an object has simple shape, because it exploits the intermediate octants results. Additionally, Chapter 2 also reviews the latest developments in SfS, such as plane sweeping, voxel colouring and space carving.

The octree construction in SfS should have the projection transforms associated with silhouette images. Thus, Chapter 3 reviews a linear projection model of a digital camera, including some useful approximation of the projection models (e.g., affine projection models). The projection is a mapping process from a point in a 3D space to a point in a 2D image domain. Thus, once a projection model is decided, the DLT algorithm estimates the linear solution of a projection model from data observations, i.e., 3D-to-2D point correspondences. Moreover, this solution can be further optimised by the LM non-linear optimisation. To facilitate the estimation of projection transforms in a multiple-view system, Chapter 3 introduces a method which relates the given knowledge of camera motion to the projection estimation. Since a SfS technique normally captures silhouette images from a circular motion, projection matrices are parameterised in terms of a rotation angle from the reference position. When some of the images in a circular motion violates the assumption of the pure rotation, the proposed method modifies them. This modification are estimated from 2D point correspondences between views. Therefore, Chapter 4 explores distinctive image features and a matching algorithm, which are particularly robust to a 3D camera motion.

It is generally assumed that a projection of a plane-like surface is well modelled as an affine transform, so that adjacent views in SfS can be related by a 2D affine transform. Thus, image matching in SfS should address an affine distortion, and as a solution for this, two image matching algorithms are proposed in Chapter 4. Both algorithms have

been developed from the Hausdorff distance which copes with cluster matching, because the proposed methods are motivated by a hypothesis that clustered features can enhance matching performance more than conventional point matching methods. The first matching method is invariant up to a similarity transform, and it extracts matching features from the distribution of detected point features (i.e., image texture is not investigated in this matching). It exploits the geometric attributes of a local point cluster, such as angles and local distances, and the result is better than other PPM algorithm. A local cluster (called a clique in this thesis) is determined by a 2D Delaunay graph, which uniquely determines a triangular graph from points. However, since a Delaunay graph is only invariant up to the similarity transform, the proposed Delaunay graph based matching cannot deal with significant affine distortion. To address this, the second matching method, called the clique descriptor matching includes affine invariant feature descriptor. The proposed clique descriptor matching searches for affine invariant regions by a MSER detector, and these regions are then described by SIFT descriptor. The proposed clique descriptor matching detects more tentative correspondences when images are affected by a 3D camera motion.

The affine invariant feature detector presented in Chapter 4 can also be used to improve the quality of an initial VH when additional images are provided. Since each additional image constructs a silhouette cone if a projection matrix is given, an initial VH can be more confined by new views of an object. However, this approach requires projection matrices for the additional new views. Moreover, these new projection matrices are defined on the same 3D world frame used in the initial reconstruction. To solve this problem, an image triplet (i.e., two images involved in an initial reconstruction and an additional new image) is investigated to calibrate a new image. In the proposed image calibration algorithm, 3D-to-2D point correspondences are located by a linear triangulation and the epipolar transfer, after the clique descriptor matching determines the 2D point correspondences between every two views. Consequently, the proposed calibration method in Chapter 5 shows how to collect point data for the calibration from the stereo reconstruction and three-view geometry.

Once the volumetric data is ready, it is better to extract surface meshes from the VH for the efficient 3D visualisation. Therefore, in Chapter 6, some fundamental surface

construction algorithms (e.g., MC, 3D convex hull, and 3D Delaunay) are explained as a literature review, and the various surface results of SfS by these methods are compared. In general, MC performs well unless there is noise in silhouettes and projection transforms. Otherwise, the surfaces extracted from SfS often fail to construct a closed surface, i.e., the resulting surface contains artefacts such as open meshes or unattached 3D segments. This is because the traditional MC algorithm premises a status of a vertex of a voxel (i.e., an inside or an intersection octant) is clearly classified. On the other hand, even when an image contains some noise, the corresponding volume data is almost intact, since the octree construction is more robust to these errors than the surface construction. Therefore, Chapter 6 proposes a robust surface construction method for non-convex object, which is especially useful when accurate modifications of silhouettes and projection transform are not available. In the proposed method, the volumetric data obtained from SfS are sliced, and the connection of clusters on each slice are estimated. This *a priori* knowledge of slice connections is exploited when clustering imperfect surface vertices locally. A local 3D point cloud defines a local convexity to which the 3D convex hull algorithm is applied. Finally, these local surfaces are combined to complete the surface construction.

7.2 Future work

A reconstruction method presented in this thesis premises that silhouette images belong to a certain motion to facilitate the projection estimation, e.g., a circular motion and an approximate circular motion. However, recent vision systems can determine a camera motion from a sequence of images. Therefore, it is possible to realise a more practical reconstruction system that adaptively selects effective images for SfS from a video clip. Furthermore, object detection and segmentation become more efficient if the motion is analysed.

The goal of this thesis is the generation of 3D object model from 2D images, but not involving texture generation which is essential for generating a photo-realistic model. Although SfS methods includes many object images, it is a challenging task to estimate true texture from them. This is because each image is captured at various illumination conditions, and the shading effects on multiple images complicate the prediction of the

true colour of a 3D point. Moreover, in order to avoid the texture of an occluded object in the current viewing direction, multiple views used in SfS should be ordered according to the viewing direction before a true texture estimation. For example, two images at 0° and 180° rotation cannot be used simultaneously, when approximating a colour of a reconstructed 3D point. Also, 3D smoothing algorithm and normal vector estimation are important for enhancing the quality of the visualisation.

Finally, it is worth exploring a matching method that can extend the proposed clique descriptor to cope with 3D object recognition. When true texture information corresponding to a 3D position is available, a new feature descriptor can be defined from the association of photometric feature with 3D geometric feature, which can enhance the performance of vision-based recognition further.

Appendix A

List of publications

1. Shin, D. and Tjahjadi, T.,"Clique descriptor of maximally stable regions" submitted to *the 12th IAPR international workshop* on structural and syntactic pattern recognition. Jun. 2008.

2. Shin, D. and Tjahjadi, T., "Similarity invariant Delaunay graph matching," submitted to *the 12th IAPR international workshop on structural and syntactic pattern recognition*, Jun. 2008.

3. Shin, D. and Tjahjadi, T., "Local hull-based surface construction from octree," *IEEE Transactions on Image Processing*, vol. 17, no. 9, Aug. 2008, pp.1251- 1260.

4. Shin, D. and Tjahjadi, T., "Triangular mesh generation of octrees of non-convex 3D objects," *The 18th International Conference on Pattern Recognition (ICPR2006)*, Aug. 2006, pp. 950-953.

5. Shin, D. and Tjahjadi, T., "3D Object reconstruction from multiple views in approximate circular motion," *Proceedings of IEEE SMC UK-RI Chapter Conference on Applied Cybernetics 2005*, Sept. 2005, pp. 70-75.

Appendix B

Preliminary projective geometry

In a perspective view, two parallel lines converge to a point, and the size of an object is scaled according to the focal length of a camera. To efficiently express these phenomenon in terms of mathematical notation, homogeneous representation has been developed in a projective space. This section introduces some preliminary projective geometry including explanations of mathematical notations of geometric primitives in a projective space.

B.0.1 Geometric primitives in 2D projective space

Suppose that an image captures a point in a 3D Euclidean space $\mathbb{R}^3$. A position of the point in an image plane is then determined by the intersection of an image plane with a ray that starts from a camera centre toward the point. A point notation should account for this pinhole camera geometry and always indicates an identical point even though an image plane changes their position under the fixed camera position. Furthermore, it should be possible for the notation to represent an image of a point at infinite called a vanishing point. A homogeneous notation describes a point $\vec{p}$ in a 2D projective space $\mathbb{P}^2$ as a vector defined by two positional elements and a scaling element, i.e., $\vec{p} = [x\ y\ s]^{\mathrm{T}}$. Therefore, a normalised homogeneous point $\bar{p}$, which has $s = 1$, is equivalent to a point $\vec{p}' = [u\ v]^{\mathrm{T}}$in $\mathbb{R}^2$, where $u = x/s$ and $u = y/s$, and zero scaling value represents a point

at infinity.

A line in $\mathbb{P}^2$ can also be described by a vector with three elements. For example, coefficients of a line equation $ax + by + c = 0$ defines a homogeneous line representation $\vec{l} = [a\ b\ c]^{\mathrm{T}}$, which becomes $\vec{l} = [ka\ kb\ kc]^{\mathrm{T}}$ when it is scaled by k. Thus, a line and a point in $\mathbb{P}^2$ are not distinguishable in the homogeneous representation, so that any theorem devised for a point in $\mathbb{P}^2$ can be interchangeable to a line and vice versa, i.e., a line is dual to a point [3]. By expressing a point and line geometry in this way, this duality is more beneficial than a traditional Euclidean vector notation. For example, an intersection point $\vec{x}$ of two lines, $\vec{l_1}$ and $\vec{l_2}$, are expressed by simple cross product of two vectors,

$$\vec{x} = \vec{l_1} \times \vec{l_2} = [\vec{l_1}]_{\times}\vec{l_2}, \tag{B.1}$$

where $[\cdot]_{\times}$ denotes a 3-by-3 skew symmetry matrix defined by 3 line coefficients with zero diagonal elements, so that $[\vec{l_1}]_{\times}^{\mathrm{T}} = -[\vec{l_1}]_{\times}$ and

$$[\vec{l_1}]_{\times} = \begin{bmatrix} 0 & -c_1 & b_1 \\ c_1 & 0 & -a_1 \\ -b_1 & a_1 & 0 \end{bmatrix}. \tag{B.2}$$

When a point $\vec{x}$ lies on a line $\vec{l_1}$, they comply with

$$\vec{x}^{\mathrm{T}}\vec{l_1} = 0, \tag{B.3}$$

and a line $\vec{l_1}$ associated with two points $\vec{x}_1$ and $\vec{x}_2$ is represented as

$$\vec{l_1} = [\vec{x}_1]_{\times}\vec{x}_2. \tag{B.4}$$

A conic is a second order 2D curve produced by the intersection of an image plane with quadrics (e.g., a sphere, a elliptical surface, and a quadratic surface defined in $\mathbb{P}^3$). Thus, a conic includes a circle, an ellipse, a hyperbola and a parabola in $\mathbb{P}^2$. Since a conic is a second order polynomial in an image plane, it is sufficient to describe a conic with six coefficients of the polynomial, which are stored in a 3-by-3 symmetric matrix. For

example, if a point $\vec{x}$ lies on a conic C, it satisfies

$$\vec{x}^{\mathrm{T}} C \vec{x} = 0, \tag{B.5}$$

where C is defined by the coefficients of $ax^2 + bxy + cy^2 + dx + ey + f = 0$, i.e.,

$$C = \begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix}. \tag{B.6}$$

Similar to the third element of a 2D point or a line, the last element f of a conic C in (B.6) becomes a scaling factor. In particular, tangent lines can also be used to define a conic, assuming the point and line duality, and this conic representation is called a dual conic.

B.0.2 Homography

When an image plane is translated by a 3D camera motion without changing a camera centre, geometric primitives are transformed between two plane spaces and this transformation in $\mathbb{P}^2$ is expressed by a non-singular 3-by-3 transform, called a homography. For example, $H\vec{x} = \vec{x}'$ denotes a point $\vec{x}'$ is transformed from $\vec{x}$ by a homography H whilst a transformed conic C' is denoted as $C' = H^{-\mathrm{T}} C H^{-1}$. A 2D homography can have up to eight Degree of Freedom (DoF) except a scaling element, so that it can be estimated from at least four point correspondences, where each pair produces two equations, between two views. Once a forward homography is estimated, the inverse transformation can be defined as a homography has full rank.

Mathematically, a general homography H is derived from the product of three essential transforms, e.g., similarity, affine and projective transform, which are classified by their DoF. For example, a homography with four DoF is sufficient to represent a

similarity transform H_{s}, i.e.,

$$H_{\mathrm{s}}(\alpha, \theta, t_{\mathrm{x}}, t_{\mathrm{y}}) = \begin{bmatrix} \alpha\cos\theta & -\alpha\sin\theta & t_{\mathrm{x}} \\ \alpha\sin\theta & \alpha\cos\theta & t_{\mathrm{y}} \\ 0 & 0 & 1 \end{bmatrix}, \tag{B.7}$$

where input parameters α, θ, and $[t_{\mathrm{x}}\ t_{\mathrm{y}}]^{\mathrm{T}}$ respectively denote uniform scaling, 2D rotation angle, and 2D translation. Therefore, angles or length ratios defined between 2D points are invariant to the similarity distortion. If a shear distortion is required, the similarity homography is modified to include two additional parameters, related to the different scaling values (λ_1,λ_2) and skewness angle ϕ. Thus, the resulting transform, called an affine homography, has 6 DoF,

$$H_{\mathrm{a}}(\theta, \phi, \lambda_1, \lambda_2, t_{\mathrm{x}}, t_{\mathrm{y}}) = \begin{bmatrix} R(\theta)R(-\phi)\mathrm{diag}(\lambda_1, \lambda_2)R(\phi) & \vec{t} \\ \vec{0} & 1 \end{bmatrix}, \tag{B.8}$$

where $R(\cdot)$ is a rotation matrix which is in the upper left 2-by-2 matrix of (B.7), and $\mathrm{diag}(\cdot)$ is a diagonal matrix whose size is dependent on the number of input parameters. A vector $\vec{t}$ in (B.8) represents a translation vector, such as $\vec{t} = [t_{\mathrm{x}}\ t_{\mathrm{y}}]^{\mathrm{T}}$. However, a vanishing point cannot be visualised in the affine geometry because parallelism is preserved under the affine distortion. To translate a vanishing point into an image plane, the last row of H_{a} should have non-zero values, which additionally creates two DoF. A resulting transform is called a projective homography H_{p}, where parallelism is no longer uphold but the invariance of concurrency and collinearity are retained.

When projecting a point onto an image plane, a point traverses three coordinates, such as 3D world coordinate, 3D camera coordinate, and 2D image plane coordinate, i.e., a point can have three representations in terms of coordinates. To avoid the confusion, all coordinates are represented with respect to the reference coordinate, called a frame, which consists of a quadrant of vectors. For example, a world frame $\mathcal{F}_w = [\vec{o}, \vec{i}, \vec{j}, \vec{k}]$ means a coordinate system with three orthonormal basis $[\vec{i}, \vec{j}, \vec{k}]$ at the origin $\vec{o}$. This thesis often stipulates a frame name as superscript of a point representation when multiple cameras are involved, e.g., $\vec{p}_i^{\,w}$ represents the i-th point in the world frame $\mathcal{F}_w$.

A projection transform, a special form of homographies that transforms a point in $\mathbb{P}^3$ onto $\mathbb{P}^2$, is defined by a 3-by-4 matrix in which the physical camera characteristics (e.g., focal length, camera centre and image skewness) are encoded as well as external 3D motion parameters (e.g., 3D rotation and translation). Suppose that P, $\vec{x}^w$ and $\vec{x}^c$ respectively represent a projection matrix, a point in $\mathbb{P}^3$ and its projection onto an image plane. The projection equation satisfies

$$\vec{x}^c = P\vec{x}^w, \tag{B.9}$$

where equality only holds up to the scale and P is called a projection matrix[1].

The analysis of a camera matrix gives practical information regarding a scene geometry. For example, the centre of a camera $\vec{c}^w$ (i.e., the origin of a camera frame $\mathcal{F}^c$) is estimated from the kernel of a projection matrix because a camera centre satisfies $P\vec{c}^w = \vec{0}$. Thus, $\vec{c}^w$ is obtained from an eigen vector of a rectangular matrix P, associated with the smallest eigenvalue, and the eigen analysis of an ill-posed matrix P is achieved by the Singular Value Decomposition (SVD) (see details of SVD in Appendix C). Then a ray produced by a back-projection can be parameterised by λ in terms of P and $\vec{c}^w$, i.e.,

$$\vec{x}^w(\lambda) = P^+\vec{x}^c + \lambda\vec{c}^w, \tag{B.10}$$

where P^+ is a pseudo-inverse of P. The four column vectors of a projection matrix also have geometrical meaning, i.e., the first three column vectors correspond to the vanishing points in the direction of three basis vectors of $\mathcal{F}^w$ and the last column represents the projection of the origin of a world frame [4]. At least six pairs of 3D-to-2D correspondences are required to linearly estimate a projection matrix consisting of 11 parameters except a scaling. Literature review of estimation techniques for a camera matrix is explained in Chapter 3.

[1] It is also known as a camera matrix and calibration matrix because camera characteristics, which is normally determined by the camera calibration process, can be derived from it (see more details in Chapter 3).

B.0.3 Two-view geometry

Epipolar geometry encodes the projective relations between two views, and provides a strong geometric constraint [5]. For example, it has been exploited as a primary restriction when searching image correspondences in stereoscopic images [6] and it can be incorporated even in a Kruppa's equation, which produces a condition for self-camera calibration [7]. When two views[2], $\vec{\pi}_1^w$ and $\vec{\pi}_2^w$, face the same scene, two epipoles of the views are defined by the intersection of the baseline that connects two camera centres $\vec{c}_1^w$ and $\vec{c}_2^w$ with two image planes $\vec{\pi}_1^w$ and $\vec{\pi}_2^w$. Thus, an image of the second camera centre $\vec{c}_2^w$ in $\vec{\pi}_1^w$ becomes an epipole $\vec{e}_2^{c1}$, where superscript $c1$ indicates an image frame $\mathcal{F}_{c1}$, and similarly $\vec{e}_1^{c2}$ is decided by $\vec{c}_1^w$. If there is a point $\vec{x}^w$ in $\mathbb{P}^3$, an epipolar plane $\vec{\pi}_x^w$ is defined by a baseline and $\vec{x}^w$. Moreover, the plane intersections $\vec{\pi}_x^w$ with $\vec{\pi}_1^w$ and $\vec{\pi}_2^w$ create epipolar lines, $\vec{l}^{c1}$ and $\vec{l}^{c2}$, on each image. In other words, a pair of point correspondences contribute to create a epipolar plane, which should encompass the same baseline, so that multiple epipolar planes establish a pencil of planes from point correspondences of two views.

This geometrical constraint is algebraically represented by a 3-by-3 singular matrix with rank 2, called a fundamental matrix. This is similar to a 2D homography except that it transforms a point in $\vec{\pi}_1^w$ to a line in $\vec{\pi}_2^w$. For example, assuming F_{12} is a fundamental matrix from $\vec{\pi}_1^w$ to $\vec{\pi}_2^w$ and there is a point correspondence $\vec{x}^{c1} \leftrightarrow \vec{x}^{c2}$, a point corresponding to x^{c1} in $\vec{\pi}_2^w$ lies on an epipolar line, i.e.,

$$\vec{l}_x^{c2} = F_{12}\vec{x}^{c1}, \tag{B.11}$$

and the transpose of a fundamental matrix defines an inverse transform, i.e., F_{12}, i.e., $F_{21} = F_{12}^{\mathrm{T}}$. By substituting (B.11)to (B.3), a fundamental matrix relates two corresponding points, i.e.,

$$(\vec{x}^{c2})^{\mathrm{T}} F_{12}\vec{x}^{c1} = 0. \tag{B.12}$$

When two projection matrices of images are known as P_1 and P_2, an epipolar line

[2] $\vec{\pi}^w$ indicates a plane in $\mathbb{P}^3$ defined from four coefficients of a plane equation, i.e., $ax + by + cz + d = 0$.

$\vec{l}_x^{c2}$ in $\vec{\pi}_2^w$ can be represented by plugging a ray equation (B.10) into (B.9), i.e.,

$$\vec{l}_x^{c2}(\lambda) = P_2 P_1^+ \vec{x}^{c1} + \lambda P_2 \vec{c}_1^w. \tag{B.13}$$

Thus, any point on an epipolar line is described by choosing an appropriate parameter λ, and by substituting two points obtained from $\lambda = 0, \infty$ into (B.4), a line can be determined without λ, i.e., $\vec{l}_x^{c2} = [P_2 \vec{c}_1^w]_\times P_2 P_1^+ \vec{x}^{c1}$.

Appendix C

Singular value decomposition

Eigen analysis of a matrix plays a significant role in linear algebra and it is an extensively exploited tool in signal processing, e.g., data compression, recognition, noise reduction and model fitting [108]. An eigen vector is defined as a vector that is invariant under transformation but it changes scale, i.e.,

$$A\vec{e} = \lambda\vec{e}, \tag{C.1}$$

where $\vec{e}$ is an eigen vector of a transform matrix A and λ denotes the corresponding eigen value of $\vec{e}$. Therefore, the eigen values corresponding to non-trivial eigenvectors are found from

$$\det(A - \lambda I) = 0. \tag{C.2}$$

Once the eigen values are determined, they are used to scale a null space of a matrix $(A - \lambda I)$, which defines the eigen vectors. However, when matrix A is a singular (e.g., over-determined, under determined or has linearly dependent row or column vectors), the determinant of $(A - \lambda I)$ is not computable. In this case, the method of Singular Value Decomposition (SVD) is used to estimate the eingen vectors and values of the rectangular matrix $A \in \mathbb{R}^{m\times n}$ by factorising of A into two orthogonal matrices (i.e., $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$) and a diagonal matrix $\Lambda \in \mathbb{R}^{m\times n}$, such as

$$A = U\Lambda V^{\mathrm{T}}. \tag{C.3}$$

The matrix diagonalisation is only applicable to a linearly independent square matrix, and SVD estimates U, V and Λ from diagonalisable square matrices derived from the inner and outer product of A. An outer product of A is

$$\begin{aligned} AA^{\mathrm{T}} &= U\Lambda V^{\mathrm{T}}(U\Lambda V^{\mathrm{T}})^{\mathrm{T}} \\ &= U\Lambda(V^{\mathrm{T}}V)\Lambda^{\mathrm{T}}U^{\mathrm{T}} \\ &\therefore (AA^{\mathrm{T}})U = U\mathrm{diag}(\lambda_1^2, \cdots, \lambda_m^2). \end{aligned} \tag{C.4}$$

Similarly, an inner product of A is derived as $(A^{\mathrm{T}}A)V = V\mathrm{diag}(\lambda_1^2, \cdots, \lambda_n^2)$. Therefore, SVD is regarded as a process that simultaneously diagonalises AA^{T} and $A^{\mathrm{T}}A$ [108]. The function $\min(m, n)$ defines the number of eigen values that are used in a rectangular matrix Λ and are padded with zeros. For example, when $m > n$, the SVD of A is

$$A_{m\times n} = U_{m\times m} \begin{bmatrix} \lambda_1 & 0 & 0 \\ & \ddots & \\ 0 & 0 & \lambda_n \\ 0 & 0 & 0 \\ & \vdots & \\ 0 & 0 & 0 \end{bmatrix}_{m\times n} V_{n\times n}. \tag{C.5}$$

Appendix D

Hausdorff distance and its variants

D.1 Traditional HD

The HD assures robustness against noise and outliers in a clique. Another advantage of HD is that it can measure distance of two sets of points without exact point correspondences. The general HD is a directional distance. If a directional HD of two point sets, $\mathcal{V}^{\mathrm{m}}$ and $\mathcal{V}^{\mathrm{t}}$, is

$$h'(\mathcal{V}^{\mathrm{m}}, \mathcal{V}^{\mathrm{t}}) = \max_{\alpha \in \mathcal{V}^{\mathrm{m}}} \min_{\beta \in \mathcal{V}^{\mathrm{t}}} \{||\vec{v}_\alpha - \vec{v}_\beta||\}, \tag{D.1}$$

where $||\cdot||$ represents the Euclidean distance, then $h'(\mathcal{V}^{\mathrm{m}}, \mathcal{V}^{\mathrm{t}}) \neq h'(\mathcal{V}^{\mathrm{t}}, \mathcal{V}^{\mathrm{m}})$. If $h'(\mathcal{V}^{\mathrm{m}}, \mathcal{V}^{\mathrm{t}})$ of two cliques is measured as ϵ, then each point of C_i^{m} must be within a distance ϵ of some point of C_j^{t} and there is also some point of C_i^{m} which is exactly a distance ϵ from the nearest point of C_j^{t} [54]. A non-directional HD is usually obtained from choosing the maximum distance of two different directional HD's, i.e.,

$$h_1(\mathcal{V}^{\mathrm{m}}, \mathcal{V}^{\mathrm{t}}) = \max(h'(\mathcal{V}^{\mathrm{m}}, \mathcal{V}^{\mathrm{t}}), h'(\mathcal{V}^{\mathrm{t}}, \mathcal{V}^{\mathrm{m}})). \tag{D.2}$$

D.2 Useful variations of HD

Variants of a non-directional HD can also be generated by changing the distant metric in (D.1) and combining two directional HD's [70]. For example, the ranked HD distance replaces max in (D.1) with the k-th largest distance, i.e.,

$$h'_k(\mathcal{V}^{\mathrm{m}}, \mathcal{V}^{\mathrm{t}}) = K_{\alpha \in \mathcal{V}^{\mathrm{m}}} \min_{\beta \in \mathcal{V}^{\mathrm{t}}} \{||\vec{v}_\alpha - \vec{v}_\beta||\}, \tag{D.3}$$

where K denotes the k-th ranked value in the set of distance (i.e., $1 \leq k \leq |\mathcal{V}^{\mathrm{m}}|$) [54]. Thus, the ranked HD is equivalent to the traditional HD when $k = 1$ and it is more robust to an outlier in the point set. However, the choice of k relies on heuristics. To avoid this problem, Dubuisson et al. propose a modified HD, which averages all distances between sets, i.e.,

$$h'_{\mathrm{m}}(\mathcal{V}^{\mathrm{m}}, \mathcal{V}^{\mathrm{t}}) = \frac{1}{|\mathcal{V}^{\mathrm{m}}|} \sum_{\alpha=1}^{|\mathcal{V}^{\mathrm{m}}|} \min_{\beta \in \mathcal{V}^{\mathrm{t}}} \{||\vec{v}_\alpha - \vec{v}_\beta||\}. \tag{D.4}$$

Some experimental results [70] show that the modified HD performs better than the ranked HD when the amount of noise is increased.

Appendix E

Robust regression

Robust regression is used to search for a model which fits to noise contaminated data. Suppose that two views are related by a projective transform, H. Thus, a point pair $\vec{p}_i^{\,\mathrm{l}} \leftrightarrow \vec{p}_i^{\,\mathrm{r}}$ satisfies the following condition,

$$||H\vec{p}_i^{\,\mathrm{r}} - p_i^{\mathrm{l}}|| = ||H^{\mathrm{T}}\vec{p}_i^{\,\mathrm{l}} - \vec{p}_i^{\,\mathrm{r}}|| = 0. \tag{E.1}$$

However, the observed data is inevitably contaminated by a noise, i.e., the observed data is

$$\hat{p}_i = \vec{p}_i + N(\vec{m}, \sigma_{\mathrm{n}}), \tag{E.2}$$

where $N(\cdot)$ denotes a Gaussian noise with mean ($\vec{m}$) and standard deviation (σ_{n}). Thus, (E.1) is revised as $||H\vec{p}_i^{\,\mathrm{r}} - p_i^{\mathrm{l}}|| = \epsilon_{\mathrm{r}}$ and the best regression of H from these Gaussian noise added data are sought by minimising the residual error, ϵ_{r}, i.e.,

$$\arg\min_{H} \left\{ \frac{1}{n} \sum_{i=0}^{n} ||\hat{p}_i^{\mathrm{r}} H - \hat{p}_i^{\mathrm{l}}|| \right\}. \tag{E.3}$$

The optimisation problem in (E.3) is generally referred to as the Least Mean Square(LMS) technique, which gives robust estimation against Gaussian like noise [106]. A closed form

of the LMS solution is also possible by rearranging (E.3) to

$$\begin{bmatrix} \vec{0}^{\mathrm{T}} & -s_i^{\mathrm{l}}(\hat{p}_i^{\mathrm{r}})^{\mathrm{T}} & y_i^{\mathrm{l}}(\hat{p}_i^{\mathrm{r}})^{\mathrm{T}} \\ -s_i^{\mathrm{l}}(\hat{p}_i^{\mathrm{r}})^{\mathrm{T}} & \vec{0}^{\mathrm{T}} & x_i^{\mathrm{l}}(\hat{p}_i^{\mathrm{r}})^{\mathrm{T}} \\ -y_i^{\mathrm{l}}(\hat{p}_i^{\mathrm{r}})^{\mathrm{T}} & x_i^{\mathrm{l}}(\hat{p}_i^{\mathrm{r}})^{\mathrm{T}} & \vec{0}^{\mathrm{T}} \end{bmatrix} \vec{h} = 0, \tag{E.4}$$

where $\vec{h}$ is a 1×9 vector vectorised from H and $\hat{p}_i^{\mathrm{l}} = [x_i^{\mathrm{l}}\ y_i^{\mathrm{l}}\ s_i^{\mathrm{l}}]^{\mathrm{T}}$ [3]. Thus, the solution exists in a null space of a matrix in (E.4).

However, when the observed data involves a point pair having severe residual error, called an outlier, LMS estimation is not reliable. One straightforward solution is to prevent these outliers when estimating a model. For example, Least Median Square (LMedS) technique randomly selects the minimum number of sample points required to estimate a model H, and computes the median value of ordered residual errors, i.e, $med(||\hat{p}_i^{\mathrm{r}}H - \hat{p}_i^{\mathrm{l}}||)$ in which $med(\cdot)$ is a function that returns the median value from sorted residual errors. Thus, the best model obtained from

$$\arg\min_{H} med(||\hat{p}_i^{\mathrm{r}}H - \hat{p}_i^{\mathrm{l}}||), \tag{E.5}$$

can classifies inliers from which the final result is sought by LMS [106].

Similarly, RANdom SAmple Consensus (RANSAC) also relies on the same strategy that randomly collects samples to avoid outliers. One difference is that RANSAC needs thresholds for the residual error and the number of inliers, which is called the consensus set in RANSAC. However, it is better than LMedS when half of data are outliers. The pseudo code of RANSAC algorithm is listed as follows.

RANSAC

```
    randomly collected sample points from an input data
    estimate a model from the sample points
    for j =1 until the end of data
        collect consensus points that is within distance threshold
    end for
    if the number of consensus points > inlier threshold
        estimate a model using LMS from consensus points
        and compute residual error of inlier points in the current model
        if previous residual error > current residual error
            replace the best model with the current estimation
        end if
    end if
end for
```

Appendix F

Programming naming conventions

The programs shown in this thesis are mainly written in C, C++, and MATLAB, and they are associated with functions from external libraries (e.g., OpenCV and OpenGL) and user-defined classes (e.g., COctant and COctree). Furthermore, the C or C++ codes are sometimes incorporated in commercial GUI wrappers (e.g., MFC in PC platform and Carbon in Macintosh platform) and even a MATLAB function utilises them as a MEX code. Therefore, to enhance code readability and reusability, most codes shown in this thesis are written to comply with a unifying naming conventions.

F.1 C++ and C

General conventions A name should succinctly describe a function, class or variable. Thus, abbreviation or acronym of words is preferable for a long name, and a capital letter is used to differentiate between words if there are more than one word in a name. Naming variables basically follows the Hungarian naming convention [107], which utilises the specific prefix to identify a data type.

Class and structure The name of a user-defined class should start with capital C followed by actual name of a class. For example, COctatnt and CImgProc represent

a class for an octant node and for an image processing, respectively. In a case that a class has multiple access modifiers, the public class elements appear first followed by private and protected elements. A member function and variable name has a prefix 'm_' to differentiate it from other local variables and functions. In C language, where a class data type is not allowed, the name of a structure data type begins with a capital character S, but member variables of a structure may not have the member prefix.

Function and local variable A name of a function which does not belong to an external library, only follows the general convention but the first word in a function name can have a capital letter. Local variables have more than one prefix to indicate its data type and characteristic (see useful prefixes in Table F.1). A member variable also has these prefixes after a member prefix m_, e.g., m_pdVertex indicates that it is a class member variable which stores vertex information using a pointer of double data type.

Special case When functions and classes are defined by a third-party library, their own naming conventions replace the general conventions. For example, Intel OpenCV functions and data types follow their conventions in [109], e.g., all OpenCV functions and data types have prefix cv or Cv. On the other hand, prefix gl or GL indicates a function and data type from the OpenGL library. More details on OpenGL naming conventions are found in [110]

F.2 MATLAB

Matrix and vector MATLAB treats all variables as a matrix. However, a scalar, a vector and a matrix variables can be distinguished by their name in order to increase code readability. For example, a single capital letter or a word which begins with capital letter represents a matrix variable, whilst a variable with a single small letter or a wold starting with a small letter represents either a scalar or a vector.

MEX file The fundamental naming conventions of a MEX file are similar to the general naming conventions of C and C++. However, as some MEX functions supplied

Table F.1: Prefix used for naming a variable

	prefix	data type	meaning
MFC and Carbon	pts	CPoint	MFC point class
	rect	CRect	MFC rectangular class
	str	CString	MFC string class
	btn	CButton	MFC button component
	edt	CEdit	MFC text edit component
	cbx	CComboBox	MFC combo box component
	dlg	CDialog	MFC dialog class
	h	HICON	MFC handle
	sz	CSize	MFC size class
	ar	CArray	MFC array class
	cid	ControlID	Carbon control structure
	k	enum	Carbon constant variable
	rect	HIRect, CGRect	Carbon rectangular structure
	pts	HIPoint, CGPoint	Carbon point structure
	sz	HISize, CGSize	Carbon size structure
	cfstr	CFString	Carbon core foundation string structure
	b	Boolean	Carbon bool data type
	r		Carbon reference structure, eg. CFStringRef
OpenCV and OpenGL	sc	CvScalar	double array with 4 elements
	img	IplImage	Intel image structure in image processing library
	pts	CvPoint	point structure
	mat	CvMat	matrix structure
	pts, pts3d	CvPoint3D32f	3D point structure
	b	GLBoolean	1 bit OpenGL boolean data type
	n	GLint	32 bit OpenGL integer data type
	f	GLFloat	32 bit OpenGL float data type
	d	GLdouble	64 bit OpenGL double data type

	prefix	data type	meaning
C and C++	n	int	32 bit integer
	f	float	32 bit float
	d	double	64 bit double
	c	char	8 bit character
	str	string	C++ string
	b	bool	1 bit boolean
	p	*	pointer
	pp	**	double pointer
	ar	vector	C++ STL vector
	s	fstream, sstream	C++ stream
	u		unsigned data type
	g		global variable
User defined Classes	mc	CCube (for MFC), CMarchingCube (for Carbon)	marching cube class
	oct	COctant	octant class
	pts3d	C3Dpoint	3D point class
	mat, rmat	CRmatrix	real number matrix class
	img	CImage	image class
	ver	SVertex	vertex structure having 3D position, colours, normal
	otr	COctree	octree class
	imgprc	CImageProc	image processing class

by MATLAB should override standard C functions (e.g, mxfree(·) replaces free(·) for dynamic memory deallocation in a MEX code), the relevant C functions should have 'mx' prefix in its name.

User-defined toolbox When a user-defined toolbox (i.e., a function library) is added to the MATLAB working environment, names of all functions in the toolbox should start with a unique prefix that can describe the purpose of the toolbox or simply the author's initial, in order to differentiate them from previously defined integrated toolboxes.

Function name and script The name of a function which is not involved in a user-defined toolbox, begins with a small letter, and a capital letter may be used between words. When a file includes cell mode operations, its name starts with a 'script' prefix.

Bibliography

[1] D. A. Forsyth and J. Ponce, *Computer vision: A mordern approach*. Upper Saddle River, US: Prentice Hall, Inc., 2003.

[2] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*. New York, US: Mcgraw-Hill, Inc., 1995.

[3] R. Hartley and A. Zisserman, *Multiple view geometry*, 1st ed. Cambridge, UK: Cambridge University Press, 2000.

[4] A. Criminisi, I. Reid, and A. Zisserman, "Single view metrology," *Int. J. Compt. Vis.*, vol. 40, no. 2, pp. 123–148, Nov. 2000.

[5] Q. T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *Int. J. Compt. Vis.*, vol. 17, no. 1, pp. 43–75, Jan. 1996.

[6] Y. Ohta and T. Kanade, "Stereo by intra-and-inter scanline search using dynamic programming," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 7, no. 2, pp. 139–154, Mar. 1985.

[7] R. Hartley, "Kruppa's equation derived from the fundamental matrix," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 2, pp. 133–135, Feb. 1997.

[8] E. Trucco and A. Verri, *Introductory techniques for 3D computer vision*, 1st ed. Upper Saddle River, US: Prentice Hall, Inc., 1998.

[9] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE J. Robot. Automat.*, vol. 3, no. 4, pp. 323–344, Aug. 1987.

[10] A. Watt, *3D computer graphics*, 3rd ed. Harlow, UK: Addison-Wesley, 2000.

[11] L. Zhang, B. Curless, and S. M. Seitz, "Rapid shape acquisition using color structured light and multi-pass dynamic programming," in *Proc. 1st IEEE Int. Symp. 3D Data Processing, Visualization, and Transmission*, 2002, pp. 24–36.

[12] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface reconstruction algorithm," in *Proc. SIGGRAPH Comput. Graphics*, vol. 21, no. 4, 1987, pp. 163–169.

[13] R. Shekhar, E. Fayyadm, R. Yagel, and J. Cornhill, "Octree-based decimation of marching cubes surface," in *Proc. IEEE Visualization*, 1996, pp. 335–342.

[14] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 2, pp. 150–162, Feb. 1994.

[15] S. Lazebnik, E. Boyer, and J. Ponce, "On computing exact visual hulls of solids bounded by smooth surfaces," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001, pp. 156–161.

[16] K.-Y. K. Wong and R. Cipolla, "Reconstruction of sclpture from its profiles with unknown camera positions," *IEEE Trans. Image Processing*, vol. 13, no. 3, pp. 381–389, Mar. 2004.

[17] W. N. Martin and J. K. Aggarwal, "Volumetric descriptions of objects from multiple views," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 5, no. 2, pp. 150–158, Mar. 1983.

[18] R. Vaillant and O. D. Faugeras, "Using extremal boundaries for 3-D object modeling," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, pp. 157–173, Feb. 1992.

[19] K.-Y. K. Wong and R. Cipolla, "Reconstruction of outdoor sculpture from silhouettes under approximate circular motion of an uncalibrated hand-held camera," *IEICE Tras. Inf. & Syst.*, vol. E87-D, no. 1, pp. 1–7, Jan. 2004.

[20] M. Potmesil, "Generating octree models of 3D objects from their silhouettes in a sequence of images," *Comput. Vis. Graph. Image Process.*, vol. 40, no. 1, pp. 1–29, Oct. 1987.

[21] R. Szeliski, "Rapid octree construction from image sequence," *Comput. Vis. Graph. Image Process.*, vol. 58, no. 1, pp. 23–32, Jul. 1993.

[22] C. H. Chien and J. K. Aggarwal, "Volume/ surface octrees for the representation of three-dimensional objects," *Comput. Vis. Graph. Image Process.*, vol. 36, no. 1, pp. 100–113, Oct. 1986.

[23] C. Xu and J. Prince, "Gradient vector flow: A new external force for snakes," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 66–71.

[24] B. Mercier and D. Meneveaux, "Shape from silhouette: Image pixels for marching cubes," *Journal of WSCG'2005*, vol. 13, no. 3, pp. 112–118, Feb. 2005.

[25] Intel Corp. (2008, August) Open source computer vision library. [Online]. Available: http://www.intel.com/technology/computing/opencv/

[26] G. Bradski, A. Kaehler, and V. Pisarevsky, "Learning-based computer vision with Intel's open source computer vision library," *Intel Tech. Journal*, vol. 9, no. 1, pp. 119–130, May 2005.

[27] T. H. Cormen, C. E. Leoserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. Cambridge, US: The MIT Press, 2003.

[28] T. hong hong and M. O. Shneier, "Describing a robot's workspace using a sequence of views from a moving camera," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 7, no. 6, pp. 721–726, Nov. 1985.

[29] H. Noborio, S. Fukuda, and S. Arimoto, "Construction of the octree approximating three dimensional objects by using multiple views," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 6, pp. 769–781, Nov. 1988.

[30] S. K. Srivastava and N. Ahuja, "Octree generation from object silhouettes in perspective views," *Comput. Vis. Graph. Image Process.*, vol. 49, no. 1, pp. 68–84, Jan. 1990.

[31] M. Adam Y, U. Yilmaz, and V. Atalay, "Silhouette-based 3D model reconstruction from multiple images," *IEEE Trans. Syst., Man, Cybern. B*, vol. 33, no. 4, pp. 582–591, Aug. 2003.

[32] N. Ahuja and J. Veenstra, "Generating octrees from object silhouettes in orthographic views," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 2, pp. 137–149, Feb. 1989.

[33] K. Shanmukh and A. K. Pujari, "Volume intersection with optimal set of directions," *Pattern Recognition Lett.*, vol. 12, no. 3, pp. 165–170, Mar. 1991.

[34] W. Niem, "Automatic reconstruction of 3D objects using a mobile camera," *Image Vision Compt.*, vol. 17, no. 2, pp. 125–134, Feb. 1999.

[35] O. Grau, T. Pullen, and G. A. Thomas, "A combined studio production system for 3-D capturing of live action and immersive actor feedback," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 3, pp. 370–380, Mar. 2004.

[36] Y.-H. Fang, H.-L. Chou, and Z. Chen, "3D shape recovery of complex objects from multiple silhouette images," *Pattern Recognition Lett.*, vol. 24, no. 9, pp. 1279–1293, Jun. 2003.

[37] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *Int. J. Compt. Vis.*, vol. 38, no. 3, pp. 199–218, Jul. 2000.

[38] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1996, pp. 358–363.

[39] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *Int. J. Compt. Vis.*, vol. 35, no. 2, pp. 151–173, Nov. 1999.

[40] K. N. Kutulakos, "Approximate N-view stereo," in *Proc. 6th European Conf. Computer Vision*, vol. 1842, 2000, pp. I:67–83.

[41] L. S. Davis, Ed., *Foundations of Image Understanding.* Norwell, US: Kluwer academic publishers, 2001.

[42] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D model construction for turn-table sequences," in *Proc. Europ. Workshop 3D Structure from Multiple Images of Large-Scale Environments*, LNCS, vol. 1506, 1998, pp. 155–170.

[43] M. Armstrong, A. Zisserman, and R. Hartley, "Self-calibration from image triplets," in *Proc. 4th European Conf. Computer Vision*, vol. 1, 1996, pp. 3–16.

[44] W. Niem, "Automatische rekonstruktion starrer dreidimensionaler objekte aus kamerabildern," PhD thesis, University of Hannover, 1999.

[45] P. Ramanathan, E. Steinbach, and B. Girod, "Silhouette-based multiple-view camera calibration," in *Proc. Vision, Modeling and Visualization*, 2000, pp. 3–10.

[46] H. Bacakoglu and M. S. Kamel, "A three-step camera calibration method," *IEEE Trans. Instrum. Meas.*, vol. 46, no. 5, pp. 1165–1172, Oct. 1997.

[47] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 10, pp. 965–980, Oct. 1992.

[48] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[49] Y. J. Abdel-Aziz and H. M. Karara, "Direct liear transformation into object space coordinates in close-range photometry," in *Proc. Close-Range Photometry*, 1971, pp. 1–18.

[50] J. Nocedal and S. Wright, *Numerical optimization*. New York, US: Springer-Verlag, 1999.

[51] W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C++: The art of scientific computing*, 2nd ed., W. H. Press and S. A. Teukolsky, Eds. Cambridge, UK: Cambridge University Press, 2002.

[52] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conf.*, 1988, pp. 147–151.

[53] D. Shin and T. Tjahjadi, "3D object reconstruction from multiple views in approximate circular motion," in *Proc. IEEE SMC UK-RI Chapter Conf. Applied Cybernetics*, 2005, pp. 70–75.

[54] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 9, pp. 850–863, Sept. 1993.

[55] A. Bowyer, "Computing Dirichlet tessellations," *Comput. J.*, vol. 24, no. 2, pp. 162–166, 1981.

[56] F. Aurenhammer, "Voronoi diagrams: A survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, Sept. 1991.

[57] G. Bebis, T. Deaconu, and M. Georgiopoulos, "Fingerprint identification using Delaunay triangulation," in *Proc. IEEE Int. Conf. Information Intell. & Systems*, 1999, pp. 452–459.

[58] D. Pedoe, *Circles: A Mathematical View*, 2nd ed. Washington DC, US: The Mathematical Association of America, 1997.

[59] A. M. Finch, R. C. Wilson, and E. R. Hancock, "Matching Delaunay graphs," *Pattern Recognit.*, vol. 30, no. 1, pp. 123–140, Jan. 1997.

[60] T. S. Caetano, T. Caelli, D. Schuurmans, and D. A. Barone, "Graphical models and point pattern matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, no. 10, pp. 1646–1663, Oct. 2006.

[61] Y. Zheng and D. Doermann, "Robust point matching for nonrigid shapes by preserving local neighborhood structures," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, no. 4, pp. 643–649, Apr. 2006.

[62] A. D. J. Cross and E. R. Hancock, "Graph matching with a dual-step EM algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 11, pp. 1236–1253, Nov. 1998.

[63] J.-D. Boissonnat and M. Teillaud, "On the randomized construction of the Delaunay tree," *Theor. Comput.*, vol. 112, no. 2, pp. 339–354, May 1993.

[64] J. O'rouke, *Computational geometry in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1998.

[65] R. N. Strickland and Z. Mao, "Computing correspondences in a sequence of non-rigid shapes," *Pattern Recognit.*, vol. 25, no. 9, pp. 901–912, Sept. 1992.

[66] L. S. Shapiro and J. M. Brady, "Feature-based correspondence: An eigenvector approach," *Image Vision Compt.*, vol. 10, no. 5, pp. 283–288, Jun. 1992.

[67] E. Saber, Y. Xu, and A. M. Tekalp, "Partial shape recognition by sub-matrix matching for partial matching guided image labeling," *Pattern Recognit.*, vol. 38, no. 10, pp. 1560–1573, Oct. 2005.

[68] M. Carcassoni and E. R. Hancock, "Spectral correspondence for point pattern matching," *Pattern Recognit.*, vol. 36, no. 1, pp. 193–204, Jan. 2003.

[69] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 4, pp. 515–519, Apr. 2004.

[70] M. P. D. Jolly and A. K. Jain, "A modified Hausdorff distance for object matching," in *Proc. 12th Int. Conf. Pattern Recognition*, 1994, pp. A:566–568.

[71] X. Yi and O. I. Camps, "Line-based recognition using a multidimensional Hausdorff distance," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 9, pp. 901–916, Sept. 1999.

[72] Y. Gao and M. K. Leung, "Face recognition using line edge map," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 6, pp. 764–779, Jun. 2002.

[73] X. Yu and M. K. Leung, "Shape recognition using curve segment Hausdorff distance," in *Proc. 18th Int. Conf. Pattern Recognition*, vol. 3, 2006, pp. 441–444.

[74] C. Gope and N. Kehtarnavaz, "Affine invariant comparison of point-sets using convex hulls and Hausdorff distances," *Pattern Recognit.*, vol. 40, no. 1, pp. 309–320, Jan. 2007.

[75] S. W. Zucker, "Relaxation labelling and the reduction of local ambiguities," in *Proc. 8th Int. Conf. Pattern Recognition*, 1976, pp. 852–861.

[76] B. Li, Q. Meng, and H. Holstein, "Similarity K-d tree method for sparse point pattern matching with underlying non-rigidity," *Pattern Recognit.*, vol. 38, no. 12, pp. 2391–2399, Dec. 2005.

[77] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Compt. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[78] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[79] T. Tuytelaars and L. J. V. Gool, "Wide baseline stereo matching based on local, affinely invariant regions," in *Proc. 11th Brit. Machine Vision Conf.*, 2000, pp. 42–56.

[80] ——, "Matching widely separated views based on affine invariant regions," *Int. J. Compt. Vis.*, vol. 59, no. 1, pp. 61–85, Aug. 2004.

[81] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," *Image Vision Compt.*, vol. 22, no. 10, pp. 761–767, Sept. 2004.

[82] P. E. Forssen, "Maximally stable colour regions for recognition and matching," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[83] M. A. Fischlerand and R. C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.* San Francisco, US: Morgan Kaufmann Publishers Inc., 1987.

[84] F. Schaffalitzky and A. Zisserman, "Viewpoint invariant texture matching and wide baseline stereo," in *Proc. 8th IEEE Int. Conf. Computer Vision*, 2001, pp. 636–643.

[85] P. E. Forssen and D. G. Lowe, "Shape descriptors for maximally stable extremal regions," in *Proc. 11th IEEE Int. Conf. Computer Vision*, 2007, pp. 1–8.

[86] O. Chum and J. Matas, "Geometric hashing with local affine frames," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 879–884.

[87] Oxford Visual Geometry Group. (2008, August) Affine covariant features. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/affine/index.html#publications

[88] B. Triggs, "Automatic calibration and the absolute quadric," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 609–614.

[89] R. Hartley, "Self-calibration of stationary cameras," *Int. J. Compt. Vis.*, vol. 22, no. 1, pp. 5–23, Feb. 1997.

[90] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *Int. J. Compt. Vis.*, vol. 8, no. 2, pp. 123–151, Aug. 1992.

[91] K.-M. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette across time part I: Theory and algorithms," *Int. J. Compt. Vis.*, vol. 62, no. 3, pp. 221–247, May 2005.

[92] K. M. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette across time part II: Applications to human modeling and markerless motion tracking," *Int. J. Compt. Vis.*, vol. 63, no. 3, pp. 225–245, Jul. 2005.

[93] C. Zhou, R. Shu, and M. S. Kankanhalli, "Handling small features in isosurface generation using marching cubes," *Comput. Graph.*, vol. 18, no. 1, pp. 845–848, Nov. 1994.

[94] K. S. Delibasis, G. M. Matsopoulos, N. A. Mouravliansky, and K. S. Nikita, "A novel and efficient implementation of the marching cubes algorithm," *Compt. Med. Imag. Grap.*, vol. 25, no. 4, pp. 343–352, Jul. 2001.

[95] T. Chen, L. Serra, and H. Ng, "Surface extraction: dividing voxels," in *Proc. 18th Int. Congress and Exhibition*, vol. 1268, Jun. 2004, pp. 225–230.

[96] I. R. Khan, M. Okuda, and S. Takahashi, "Regular 3D mesh reconstruction based on cylindrical mapping," in *Proc. IEEE Int. Conf. Multimedia and Expo*, vol. 1, 2004, pp. 27–30.

[97] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature-sensitive surface extraction from volume data," in *Proc. SIGGRAPH Comput. Graphics*, 2001, pp. 57–66.

[98] Y. Yemez and F. Schmitt, "3D reconstruction of real objects with high resolution shape and texture," *Image Vision Compt.*, vol. 22, no. 13, pp. 1137–1153, Nov. 2004.

[99] L. P. Chew, "Constrained Delaunay triangulations," *Algorithmica*, vol. 4, no. 1, pp. 97–108, Jun. 1989.

[100] J. Ruppert and R. Seidel, "On the difficulty of triangulating three-dimensional nonconvex polyhedra." *Discrite Comput. Geom*, vol. 7, no. 3, pp. 227–253, Dec. 1992.

[101] Y.-J. Yang, J.-H. Young, and J. guang Sun, "An algorithm for tetrahedral mesh generation based on conforming constrained Delaunay tetrahedralization," *Comput. Graph.*, vol. 29, no. 4, pp. 606–615, Jul. 2005.

[102] Q. Du and D. Wang, "Boundary recovery for three dimensional conforming Delaunay triangulation," *Comput. Method. Appl. M.*, vol. 193, no. 23-26, pp. 2547–2563, Jun. 2004.

[103] D. Shin and T. Tjahjadi, "Triangular mesh generation of octrees of non-convex 3D objects," in *Proc. 18th Int. Conf. Pattern Recognition*, 2006.

[104] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. New York, US: John Wiley and Sons, Inc., 2000.

[105] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw*, vol. 22, no. 4, pp. 469–483, Dec. 1996.

[106] P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim, "Robust regression methods for computer vision: A review," *Int. J. Compt. Vis.*, vol. 6, no. 1, pp. 59–70, Apr. 1991.

[107] H. M. Deitel and P. J. Deitel, *C++ how to program*, 4th ed. Upper Saddle River, US: Prentice Hall, Inc., 2003.

[108] T. K. Moon and W. C. Stirling, *Mathematical methods and algorithms for signal processing*. Upper Saddle River, US: Prentice Hall, Inc., 1999.

[109] (2008, August) OpenCV wiki. [Online]. Available: http://opencvlibrary.sourceforge.net/

[110] D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL programming guide: The official guide to learning OpenGL*, 6th ed. Harlow, UK: Addison-Wesley, 2007.